# DIDACTICIEL D'INITIATION À L'ENVIRONNEMENT DE CONCEPTION FPGA

**Systèmes logiques programmables**

**ISE12.4, Nexys3, Isim**







École Polytechnique de Montréal - Université de Lille 1

Version 2.1 (Automne 2012)

## Version du document

| Version | Description | Auteurs | Date |
|---|---|---|---|
| 1.0 | Version originale | Benoit Gosselin | Été 2009 |
| 1.1 | Modification de la procédure de programmation du PROM SPI embarqué/d'expansion. Élimination de la compilation des librairies Xilinx avec ModelSim. | Sébastien Ethier | Automne 2009 |
| 1.2 | Correction de la procédure de programmation du PROM SPI embarqué/d'expansion. Ajout de la configuration des simulations de ModelSim. | Sébastien Ethier | Automne 2009 |
| 1.3 | Correction des compteurs de l'annexe 1 | Nicolas Laflamme-Mayer | Hiver 2011 |
| 1.4 | Correction de la MSA en annexe 5 (état c et e) | Jérôme Le Lan | Hiver 2012 |
| 2.0 | Mise à jour pour ISE 12.4<br>Adaptation à l carte Nexys2<br>Utilisation de Isim | Jean-Luc Dekeyser | Eté 2012 |
| 2.1 | Pour la carte Nexys 3 ISE 12.4 | Jean-luc Dekeyser | Automne 2012 |

## À propos de ce didacticiel

Ce didacticiel propose un survol de l'outil intégré de conception ISE version 12.4 de Xilinx et de la carte de développement Nexys3 de Digilent (http://www.digilentinc.com/). Les connaissances acquises à l'aide de ce matériel pédagogique serviront à effectuer les séances de travaux pratiques prévus dans le cadre du cours.

Documentions supplémentaire recommandée :

- Manuel d'utilisation de la carte Nexys3
  (http://www.digilentinc.com/Data/Products/NEXYS3/Nexys3_rm.pdf)
- Fiche technique de la carte Nexys3
- (http://www.digilentinc.com/Data/Products/NEXYS3/NEXYS3_sch.pdf)
- Manuel d'utilisation ISE version 12.4
  (http://www.xilinx.com/support/documentation/dt_ise12-4_userguides.htm)
- Manuel d'utilisation de Isim
  (http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/plugin_ism.pdf)

Didacticiels Xilinx recommandés :

- ISE In-Depth Tutorial version 12.4
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_4/ise_tutorial_ug695.pdf
- Isim In-Depth Tutorial
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/ug682.pdf

De plus, le site web de la compagnie Xilinx (www.xilinx.com) regorge d'informations et d'exemples sur le matériel utilisé dans ce cours. Le site comporte aussi un FAQ et une banque de problèmes résolus. N'hésitez pas à le consulter.

# Table des matières

# Lexique

| | |
|---|---|
| Flow de conception | Ensembles étapes et ordre dans lequel les réaliser pour compléter l'étape de conception du projet. |
| Instance top level | Instance de niveau hiérarchique supérieur dans le projet. C'est cette instance ou fichier qui regroupe tous les composants de niveau inférieur du projet. |
| Langage HDL | Langage de description matérielle (hardware description language) |
| Langage VHDL | VHSIC (Very High Speed Integrated Circuits) hardware description language. |
| Module DCM | Modules spécialisés dans la gestion d'horloge disponibles dans le FPGA Spartan 3E. |
| Module IP | Composant qui est disponible sous forme de black box dans le projet. ISE permet de générer des IP automatiquement grâce à un Wizard. |
| Outils EDA | Outil de conception assistée par ordinateur (Electronic design automation). |
| Design rule check (DRC) | Outil qui permet de vérifier la validité d'un schéma ou d'une implémentation à l'aide d'une liste de règles en vigueur. |
| ISE Project Wizard | Boîte de dialogue utilisée dans ISE pour créer et configurer un nouveau projet. |
| ISE Source Wizard | Boîte de dialogue utilisée dans ISE pour créer et configurer un nouvel élément du projet. |
| ISE Architecture Wizard | Boîte de dialogue utilisée dans ISE pour configurer les ressources d'un composant programmable dont des DCM, mémoire RAM, etc. |
| CORE Generator | Interface permettant de générer de configurer des composants IP. |

## Survol de Navigateur de Projet ISE

La Figure 1 identifie les principaux éléments de l'interface du Navigateur de Projet ISE utilisé dans ce didacticiel. Pour installer ce logiciel sur les machines de TP :

1- Récupérer le fichier de licence où l'on vous l'indique et l'enregistrer sur votre compte

2- Depuis un terminal, lancer le gestionnaire de licences par la commande:

/opt/Xilinx/12.4/ISE_DS/common/bin/lin/xlcm

3- Dans l'interface de gestion de licences choisir "Get free webpack licence"

4- Cliquer sur "connect now" puis sur "copy licence"

5- Charger le fichier de licence puis cliquer sur "close"

Remarque: les étapes de 1 à 5 ci-dessus sont à réaliser par chacun sur son propre compte (une seule fois pour tout le semestre)

6-Le logiciel ISE est maintenant prêt à être utiliser. Il se lance avec la commande:

/opt/Xilinx/12.4/ISE_DS/ISE/bin/lin/ise

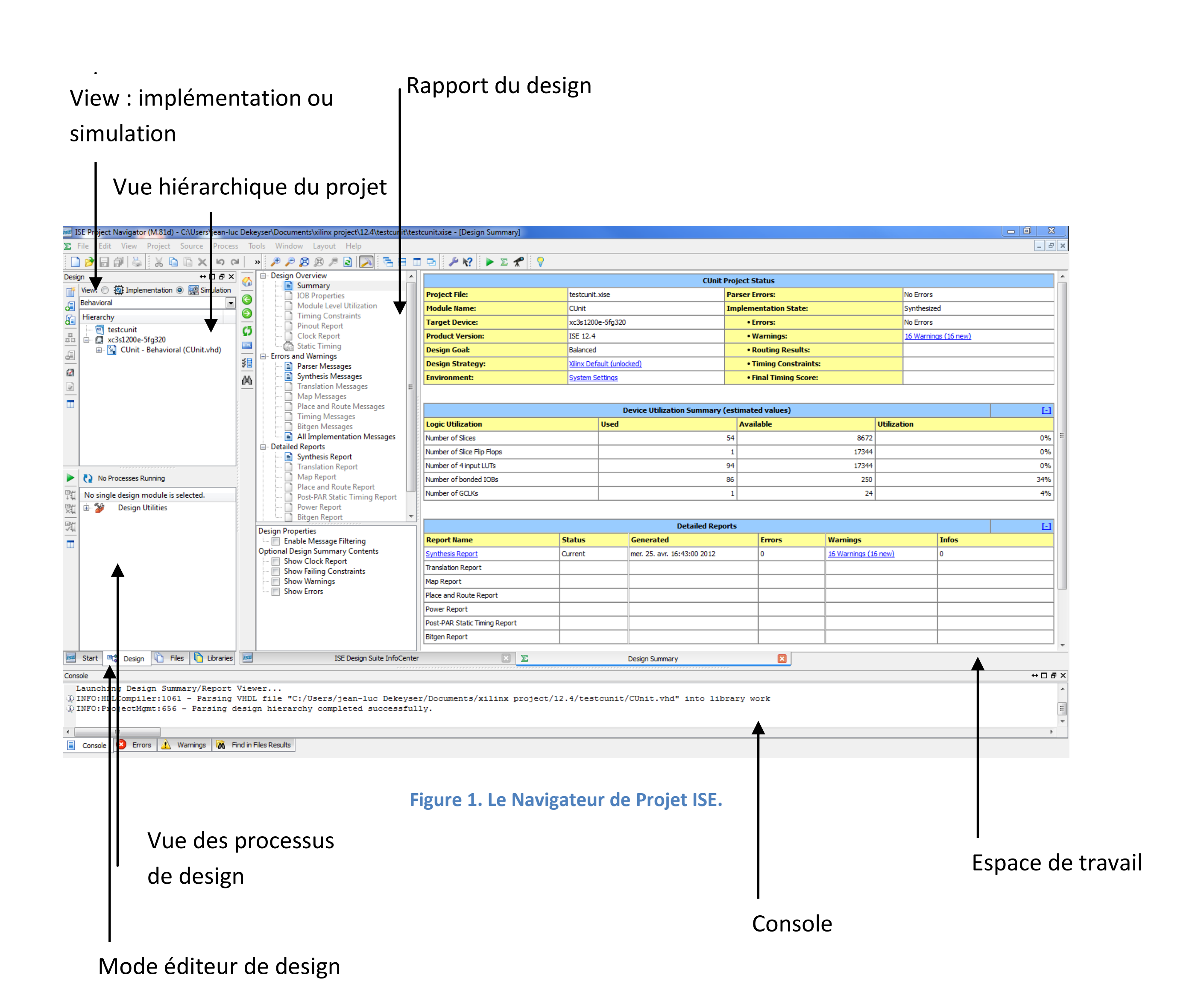



**Figure 1. Le Navigateur de Projet ISE.**

## Introduction

Ce didacticiel effectue un survol des caractéristiques principales de la carte FPGA Nexys3 à base de Spartan 6, de la suite de conception ISE, des logiciels de simulation et de synthèse disponibles dans les salles TP ainsi que du flow de conception FPGA en général.

Après ce didacticiel, vous posséderez les bases de deux approches de conception FPGA (conception par schéma et par langage HDL), vous serez en mesure d'utiliser les ressources principales de la carte Nexys3 et vous pourrez configurer la carte avec un projet. Ces connaissances vous permettront d'effectuer les travaux pratiques prévus lors de ce cours. Différentes ressources de la carte Nexys3 seront utilisées lors de chaque projet et le niveau de difficulté augmentera à chaque fois.

Dans ce didacticiel, vous réaliserez un détecteur de séquence synchrone. Le détecteur à concevoir permet d'identifier une séquence entrée par un usager à l'aide de trois boutons poussoir. Les boutons sont activés à tour de rôle dans le bon ordre pour générer la séquence. Lorsque le système détermine que la bonne séquence a été entrée, il active une série de diodes électroluminescentes (Led).

Ce projet vous familiarisera avec plusieurs composants fondamentaux des systèmes logiques dont les MSA (machines séquentielles algorithmique ou circuits logiques séquentiels) on utilisera le terme MSA dans la suite ce document, les registres à décalage, les anti-rebonds, les modules de gestion d'horloge, les mémoires et les différents types de circuits combinatoires. Vous réutiliserez les composants conçus et les connaissances acquises lors de didacticiel à plusieurs reprises lors des travaux pratiques. Enfin, ce didacticiel vous mettra en contact avec les ressources de la carte Nexys3 à base de Spartan6. En effet, l'interface avec le détecteur utilisera l'horloge de référence à 100 MHz, les boutons poussoir, les Led, les mémoires ROM embarquées ainsi que le port de configuration USB.

## Survol de l'environnement de conception

Le Navigateur de projet ISE sera utilisé comme outil de conception par excellence lors de ce didacticiel ainsi que lors des travaux pratiques. Cet outil de Xilinx permet de créer des projets comportant plusieurs types de fichiers (HDL, schématique, UCF, EDIF, etc.), de compiler, d'effectuer des design rule check (DRC), de créer des contraintes d'implémentation dont des contrainte de timings sur les horloges, de déterminer l'emplacement des broches, de créer des bancs d'essai de simulation (testbench) et de gérer efficacement les projets d'envergure.

Le Navigateur de projet ISE offre un environnement de conception centralisé extrêmement efficace qui regroupe tous les outils nécessaires à la conception, la simulation et à l'implémentation d'un projet d'envergure ainsi qu'à la configuration de la carte, la nexys3 en ce qui nous concerne.

## Carte de développement Nexys3

La carte de développement Nexys3 de Digilent utilise le FPGA Spartan (XC6LX16) de Xilinx. Cette carte offre un environnement de conception très adapté pour le prototypage d'applications variées dont celles des systèmes numériques à usage général et des systèmes embarqués. Cette carte est de plus idéale pour les applications de traitement vidéo et de traitement de signal en général.

La carte Nexys3 regroupe entre autre un FPGA XC6LX16 Spartan 6, un accès USB2 pour la configuration et le transfert de données rapide (On pourra utiliser le logiciel Adept http://digilentinc.com/Data/Documents/Tutorials/Adept%20Software%20Basic%20Tutorial.pdf).

Caractéristiques principales:

- Base de développement FPGA complète avec câble de programmation "USB" livré
- Equipé d'un FPGA Spartan-6™ (XC6LX16-CS324)
- 48 MB répartis en: 16 MB RAM + 16 MB PCM + 16 MB SPI PCM
- Ethernet 10/100 SMSC LAN8710 PHY via RJ45
- Régulateur à découpage intégrée
- Oscillateur 100 MHz
- 4 connecteurs d'extensions pour utilisation de modules optionnels "PMOD" (voir au bas de la page):
- Sortie port VGA 8 bit
- Sortie port USB (par exemple pour clavier, souris ou clef USB, non livrés)
- Mise à disposition d'un connecteur VHDC
- 4 afficheurs 7 segments à Leds
- 8 Leds
- 8 interrupteurs et 5 boutons-poussoir

Les ports d'E/S du FPGA sont repris sur 4 connecteurs femelles spécifiques. Ces connecteurs permettent l'utilisation de petits modules d'extensions optionnels (appelés "Pmod"). Une fois enfichées sur les connecteurs femelles de la platine "NEXYS3", ces modules vous permettrons

d'adjoindre de multiples possibilités et interfaces supplémentaires. La platine "NEXYS3" dispose également d'un connecteur d'extension spécial dédié aux signaux haute vitesse . Ce connecteur est destiné à recevoir diverses platines d'extension "Vmod™".

Le FPGA peut être programmé de deux façons: directement à partir d'un PC en utilisant le port USB, et à partir de la  Flash ROM (la Flash ROM est également programmable par l'utilisateur via le Port USB). Un cavalier sur la carte Nexys3 détermine la source (PC ou ROM) du FPGA à utiliser pour charger sa configuration. Le FPGA peut automatiquement charger une configuration à partir de la ROM Flash Platform à la mise sous tension si le cavalier de mode de configuration est réglé sur "Master série ". Si le cavalier Mode est réglé sur "JTAG", le FPGA attendra la programmation à partir du PC (via le câble USB). C’est cette configuration qu’on utilisera lors des TP. Les cavaliers sur les cartes sont normalement dans les bonnes configurations.

Le logiciel Adept peut être utilisé pour configurer le FPGA avec n'importe quel fichier approprié stocké sur l'ordinateur. Adept utilise le câble USB pour transférer un fichier binaire sélectionné à partir du PC vers le FPGA ou la Flash ROM. Alors le FPGA est configuré, il le restera ainsi jusqu'à ce qu'il soit remis à zéro par une rupture d'alimentation ou par une pression sur le bouton de réinitialisation du FPGA (BTNR). La Flash ROM conservera le fichier binaire jusqu'à ce qu'elle soit reprogrammée, indépendamment de la mise sous tension.

Pour une description approfondie de la carte Nexys3, consultez le manuel d’utilisation ainsi que la fiche technique disponible chez DIGILENT. N’hésitez pas à consulter ces documents pour vous aider lors des travaux pratiques. Référez-vous à la Figure 2 pour une vue d’ensemble de la carte Nexys2 et ses principaux modules.

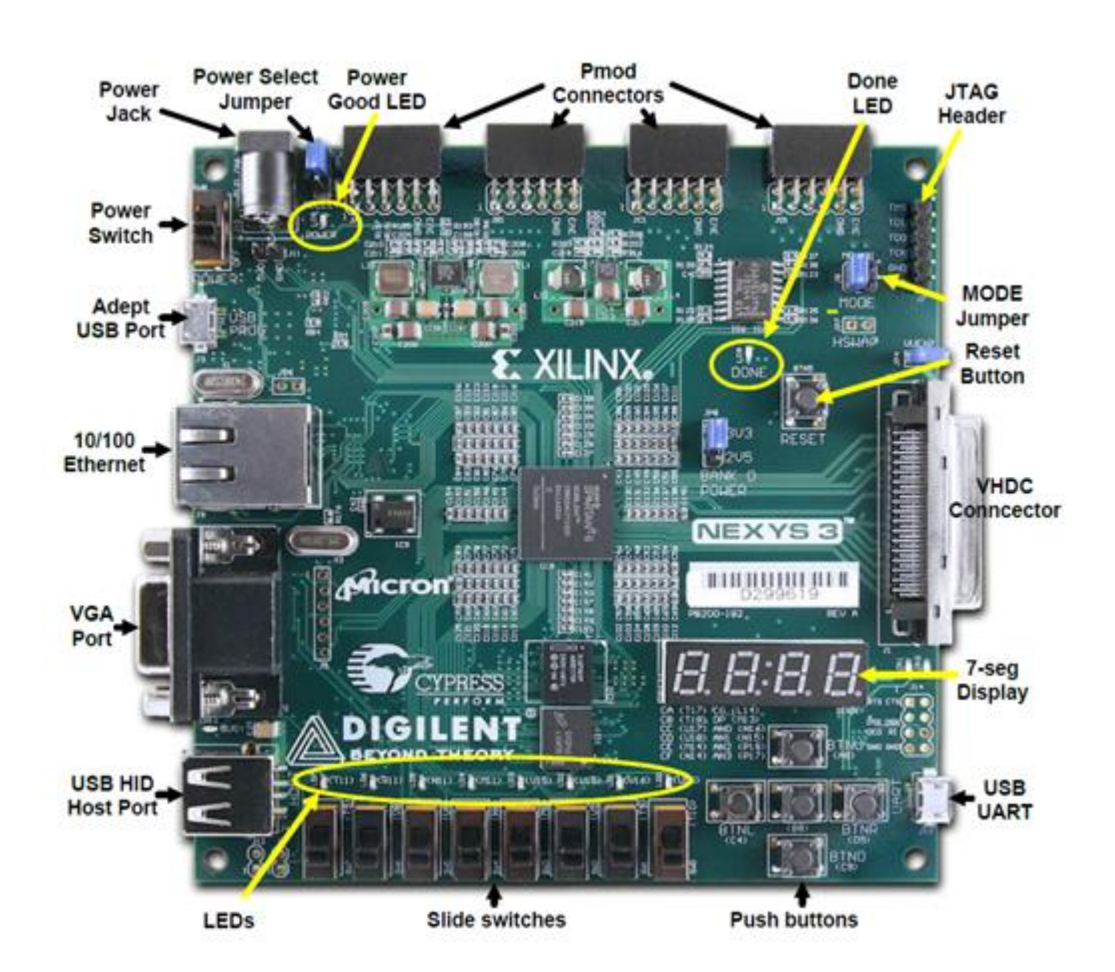



**Figure 2. La carte Nexys3 et ses principaux modules embarqués.**

## Logiciels et outils de conception

Les différents outils de conception que vous utiliserez lors des travaux pratiques sont tous intégrés et accessibles par l'entremise du Navigateur de projet ISE. ISE permet une grande efficacité car il regroupe plusieurs aspects de la conception d'un projet dans un seul environnement et un seul jeu de fichiers.

Lors de ce didacticiel, vous utiliserez l'outil de conception et simulation Isim pour effectuer les simulations fonctionnelles et les simulations avec timings de vos projets. Vous utiliserez l'outil XST de Xilinx pour effectuer la synthèse des différents projets. Notez que pour ce qui est de l'étape de synthèse, seule l'utilisation de l'outil XST est vue dans ce didacticiel. Enfin, les outils PlanAhead et ADEPT de Xilinx et DIGILENT seront utilisés respectivement pour implémenter le projet (placement

et routage) et configurer les composants de la carte avec le projet implémenté (FPGA, CPLD, mémoire ROM, etc.).

# Description de la méthodologie de conception

Ce didacticiel couvre deux approches de conception FGPA : l'approche de conception par schéma et l'approche de conception par langage HDL. Bien qu'assez différentes, les deux approches mènent exactement au même résultat final. Ainsi, les circuits composant le détecteur de séquence seront réalisés successivement avec les deux approches dans un but pédagogique. Cela vous permettra de voir les similitudes entre les deux réalisations. Vous constaterez que l'approche par schéma est plus près du schéma bloc et de la réalisation du circuit alors que l'approche par langage HDL est liée davantage aux processus algorithmiques et semble par conséquent plus abstraite.

Par ailleurs, l'approche par schéma est sans doute plus instinctive que l'approche HDL. Elle est d'ailleurs davantage préconisée par les concepteurs débutants ou novices. Les concepteurs aguerris préfèrent généralement l'approche par langage HDL parce qu'elle est plus flexible et plus efficace. De plus, cette dernière présente une meilleure compatibilité avec les outils EDA et elle est davantage employée dans l'industrie.

## Flow de conception FPGA

La Figure 3 illustre le flow de conception FPGA mis de l'avant dans ce didacticiel à l'aide d'un schéma.

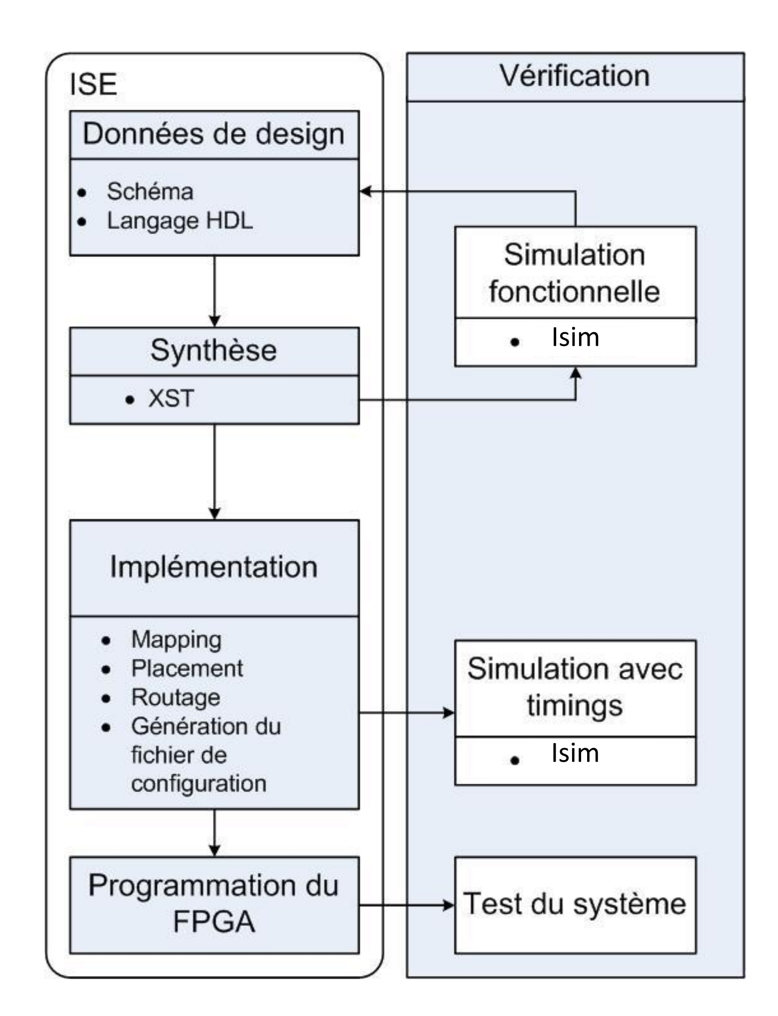



Figure 3. Flow de conception adopté dans ce didacticiel.

## Description du système à concevoir

Dans ce didacticiel, vous concevrez un détecteur de séquence synchrone. Ce système permet de détecter une séquence entrée par l'usager à l'aide de trois boutons poussoir. Les boutons doivent être activés à tour de rôle dans le bon ordre pour générer la bonne séquence. Lorsque le système détermine que la bonne séquence a été entrée, il active une série de diodes électroluminescentes (Led).

Les ressources de carte qui seront utilisées pour activer les entrées/sorties du détecteur de séquence sont les suivantes : Les cinqboutons poussoir (BTNS, BTNL, BTNR, BTNU, BTND) et les huit Leds LD7 à LD0. Les boutons doivent être activés dans l'ordre suivant pour générer la bonne séquence : BTNL, BTNR, BTNS.

**Note :** La carte Nexys3 ne dispose pas de circuits anti-rebond pour traiter les boutons poussoir. Il sera donc primordial d'éliminer les rebonds à l'aide d'un système conçu dans le FPGA. Ce didacticiel démontrera entre autres un système d'anti-rebond que vous pourrez réutiliser lors des trois travaux pratiques.

### Schéma bloc du détecteur de séquence synchrone

La Figure 4 présente le schéma bloc du détecteur de séquence que vous concevrez dans ce didacticiel. Notez l'utilisation d'un module DCM et d'un diviseur d'horloge pour dériver des signaux d'horloge de fréquence moindre, un système d'anti-rebond pour traiter les signaux d'entrée activés par les boutons poussoirs, une MSA pour détecter la séquence d'évènements ainsi que d'un registre à décalage pour activer une série de Led lorsque la séquence sera détectée.

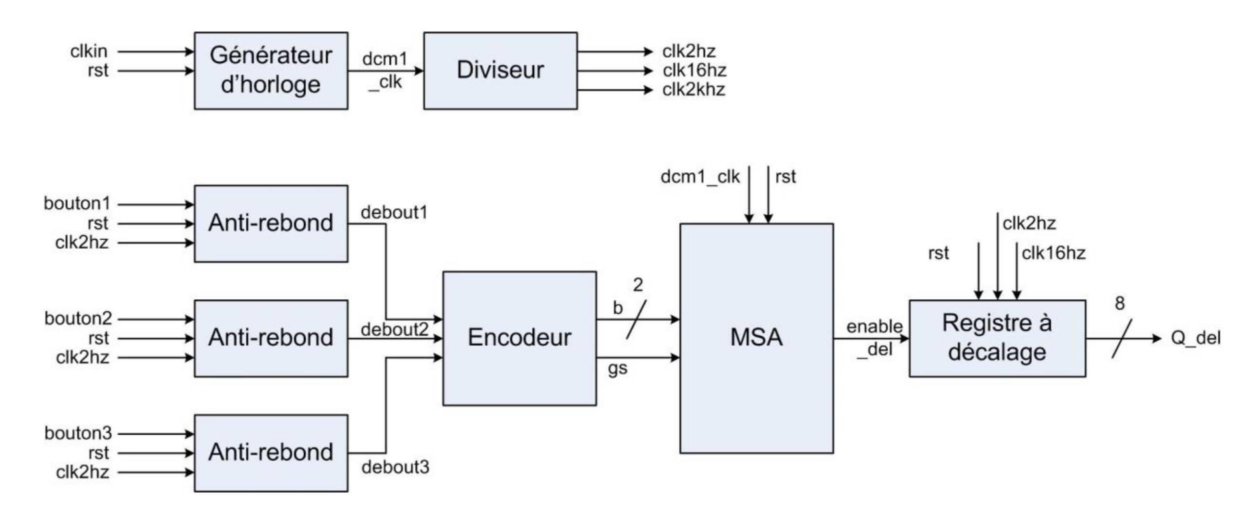



Figure 4. Schéma bloc du détecteur de séquence synchrone à concevoir.

## Diagramme d'état de la MSA

L'utilisation d'une MSA est tout indiquée pour détecter une séquence d'évènements. Ce projet utilise une MSA à cinq états dont le diagramme est montré à la Figure 5.

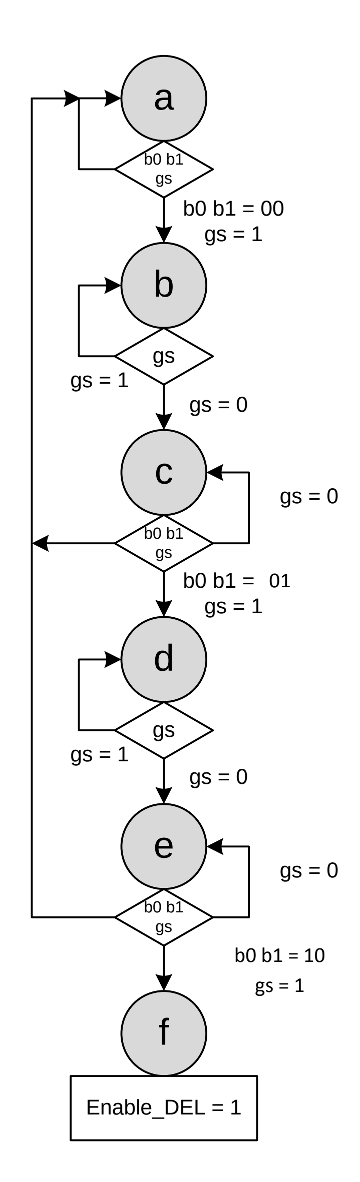



**Figure 5. Diagramme d'état de la MSA du détecteur de séquence.**

**Tableau 1. Description des états de la MSA du détecteur de séquence.**

| État | Description |
|---|---|
| a | État initial. Passe à b lorsqu'on entre b1b0 = 00 et gs = 1 |
| b | État de blocage pour traiter l'entrée activée par le bouton poussoir convenablement. On sort de cet état quand le bouton est relâché, soit quand gs repasse à 0. |
| c | Passe à d lorsqu'on entre b1b0 = 01 et gs = 1 |

| | |
|---|---|
| d | État de blocage pour traiter l'entrée activée par le bouton poussoir.<br>On sort de cet état quand le bouton est relâché, soit quand gs repasse à 0. |
| e | Passe à f lorsqu'on entre b1b0 = 10 et gs = 1 |
| f | État final. La sortie enable_del est activée. |

## Schéma bloc de la MSA

Une MSA est composée 1) d'un circuit combinatoire IFL (input forming logic) pour traiter les entrées et générer les états suivants, 2) d'un registre d'état pour maintenir la valeur de l'état courant pendant toute la durée d'une période d'horloge et 3) d'un circuit combinatoire OFL (output forming logic) pour activer les sorties de la MSA. La Figure 6 montre le schéma bloc de la MSA utilisée dans le détecteur de séquence. Cette structure sera successivement implémentée à l'aide des deux approches de conception préconisées dans ce didacticiel, soit l'approche de conception par schéma et l'approche de conception par langage HDL.

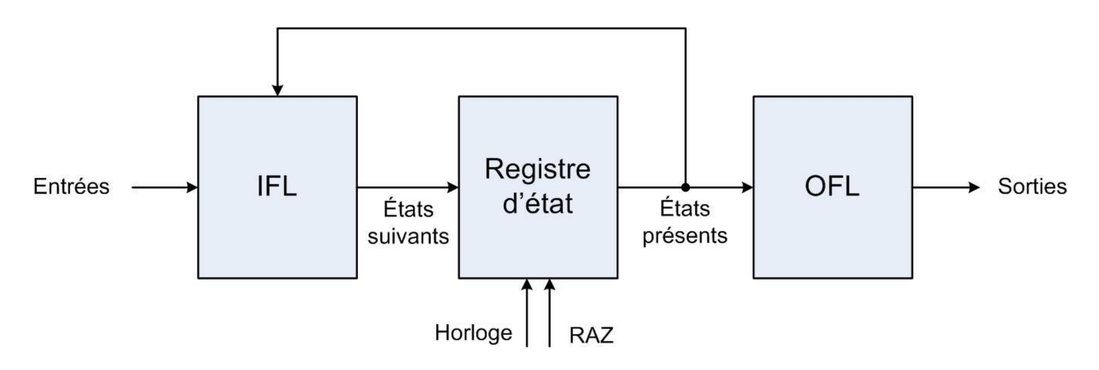



Figure 6. Schéma bloc d'une MSA.

Les tableaux suivants présentent des descriptions détaillées des entrées/sorties et des signaux internes du détecteur de séquence et propose un survol de ses composants principaux.

Tableau 2. Description des entrées/sorties.

| Nom | E/S | S/A | Taille | Description |
|---|---|---|---|---|
| rst | E | A | 1 | Signal de remise à zéro du système. Ce signal est activé par le bouton poussoir BTND de la carte Nexys3. |
| clkin | E | S | 1 | Horloge de référence du système à 100 MHz. Ce signal |

| | | | | |
|---|---|---|---|---|
| | | | | provient de l'oscillateur installé sur carte Nexys3( V10). |
| bouton1 | E | A | 1 | Ce signal est activé par un bouton poussoir de la carte Nexys3. C'est le premier bouton de la séquence à détecter. |
| bouton2 | E | A | 1 | Ce signal est activé par un bouton poussoir de la carte Nexys3. C'est le deuxième bouton de la séquence à détecter. |
| bouton3 | E | A | 1 | Ce signal est activé par un bouton poussoir de la carte Nexys3. C'est le troisième bouton de la séquence à détecter. |
| Q_del | S | S | 8 | Chaque signal de ce bus active une diode (LD7 à LD0) de la carte Nexys3 |

E/S : Entrées/sorties

S/A : Synchrone/asynchrone

**Tableau 3. Description des composants.**

| **Nom de fichier*** | **Identificateur d'instance** | **Description** |
|---|---|---|
| dcm1 | dcm1_inst | Module de gestion d'horloge. Prend l'horloge de référence à 100 MHz et génère une horloge dcm1_clk dérivée à 16 MHz, d'un rapport cyclique de 50%. Ce module est un IP instancié avec CORE Generator. |
| diviseur_clk | diviseur_inst | Module VHDL diviseur d'horloge. Ce module génère les signaux clk2hz, clk16hz, clk2khz tous trois dérivés de dcm1_clk. |
| debounce_sc<br>debounce_hdl | debounce1..3_inst | Ce module sert à éliminer les oscillations et le bruit mécanique (les rebonds) liés à l'activation d'un commutateur. Le système utilise trois instances de ce module pour traiter les trois boutons de la séquence. |
| encodeur_sc<br>encodeur_hdl | encodeur_inst | Ce module sert à encoder les trois signaux entré individuellement activés par les boutons. Il est réalisé avec deux LUT. |
| msa_sc<br>msa_hdl | msa_inst | Ce module réalise la MSA qui sert à reconnaître la séquence entrée à l'aide des trois boutons. Il est composé d'une ROM implémentant l'IFL et l'OFL et d'un registre de 3 bascules D réalisant le registre d'état. La ROM est réalisée à l'aide de CORE Generator. |
| SR8CE<br>(Utilisé uniquement | shiftreg_inst | Registre à décalage comportant 8 bits dérivé d'une LUT du Spartan 6. Il sert à activer les LED de façon séquentielle. |

| | | |
|---|---|---|
| pour l'approche par schéma) | | |

*Le suffixe sc est utilisé pour désigner les fichiers utilisés avec l'approche de conception par schéma tandis que le suffixe hdl est utilisé pour les fichiers de l'approche de conception par langage HDL.

**Tableau 4. Description des signaux internes.**

| Nom | Taille | Description |
|---|---|---|
| clk_dcm1 | 1 | Signal d'horloge à 16 MHz. Sortie du module de gestion d'horloge dcm_inst. Son rapport cyclique est de 50%. |
| clk2hz | 1 | Signal carré d'une fréquence de 2 Hz dérivé de clk_dcm1 ($f_{clk_dcm1}$ ÷ 8e6). Son rapport cyclique est de 50%. |
| clk16hz | 1 | Signal d'horloge d'une fréquence de 16 Hz dérivé de clk_dcm1 ($f_{clk_dcm1}$ ÷ 1e6). Son rapport cyclique est de 50%. |
| clk2khz | 1 | Signal d'horloge d'une fréquence de 2 kHz dérivé de clk_dcm1 ($f_{clk_dcm1}$ ÷ 8e3). Son rapport cyclique est de 50%. |
| deb_out1 | 1 | Sortie du module d'anti-rebond pour BTNR(actif haut). |
| deb_out2 | 1 | Sortie du module d'anti-rebond pour BTNS (actif haut). |
| deb_out3 | 1 | Sortie du module d'anti-rebond pour BTNL (actif haut). |
| b0 | 1 | Bit de sortie de l'encodeur/entrée de la MSA. Ce signal encode les trois entrées commandées par les boutons. |
| b1 | 1 | Bit de sortie de l'encodeur/entrée de la MSA. Ce signal encode les trois entrées commandées par les boutons. |
| gs | 1 | Ce signal est activé quand l'usager appuie sur un des trois boutons (actif haut). |
| enable_del | 1 | Ce signal est activé quand l'usager appui sur les boutons dans la bonne séquence (actif haut). |
| etatpres* | 3 | Ce bus donne la sortie du registre d'état de la MSA. Il représente l'état présent de la MSA et peut être utilisé à des fins de test lors d'une simulation. |
| etatsuiv* | 3 | Ce bus donne la sortie de l'IFL de la MSA. Il représente l'état suivant de la MSA et peut être utilisé à des fins de test lors d'une simulation. |

*Ces signaux internes sont utilisés uniquement pour l'approche de conception par schéma.

## Approche de conception par schéma

La première partie de ce didacticiel porte sur la conception du détecteur de séquence à l'aide de l'approche de conception FPGA par schéma. Dans cette partie, le projet du détecteur de séquence présente une structure hiérarchique reposant sur des modules schématiques. En effet, le fichier top level du projet contient une représentation schématique de l'ensemble du système qui fait appel à plusieurs types macros de niveaux inférieurs. Ces macros utilisent plusieurs types de modules de natures variées dont des modules schématiques, des IP générés à l'aide de l'Architecture Wizard, des modules paramétrés générés à l'aide du CORE Generator de ISE et des modules VHDL.

La figure suivante présente le schéma complété du détecteur de séquence synchrone. Au terme de cette section vous devrez obtenir un résultat conforme à cette représentation pour être en mesure de poursuivre avec les étapes subséquentes du didacticiel, soit la simulation fonctionnelle, l'implémentation du projet, la simulation avec timings et la configuration du FPGA avec le système conçu.

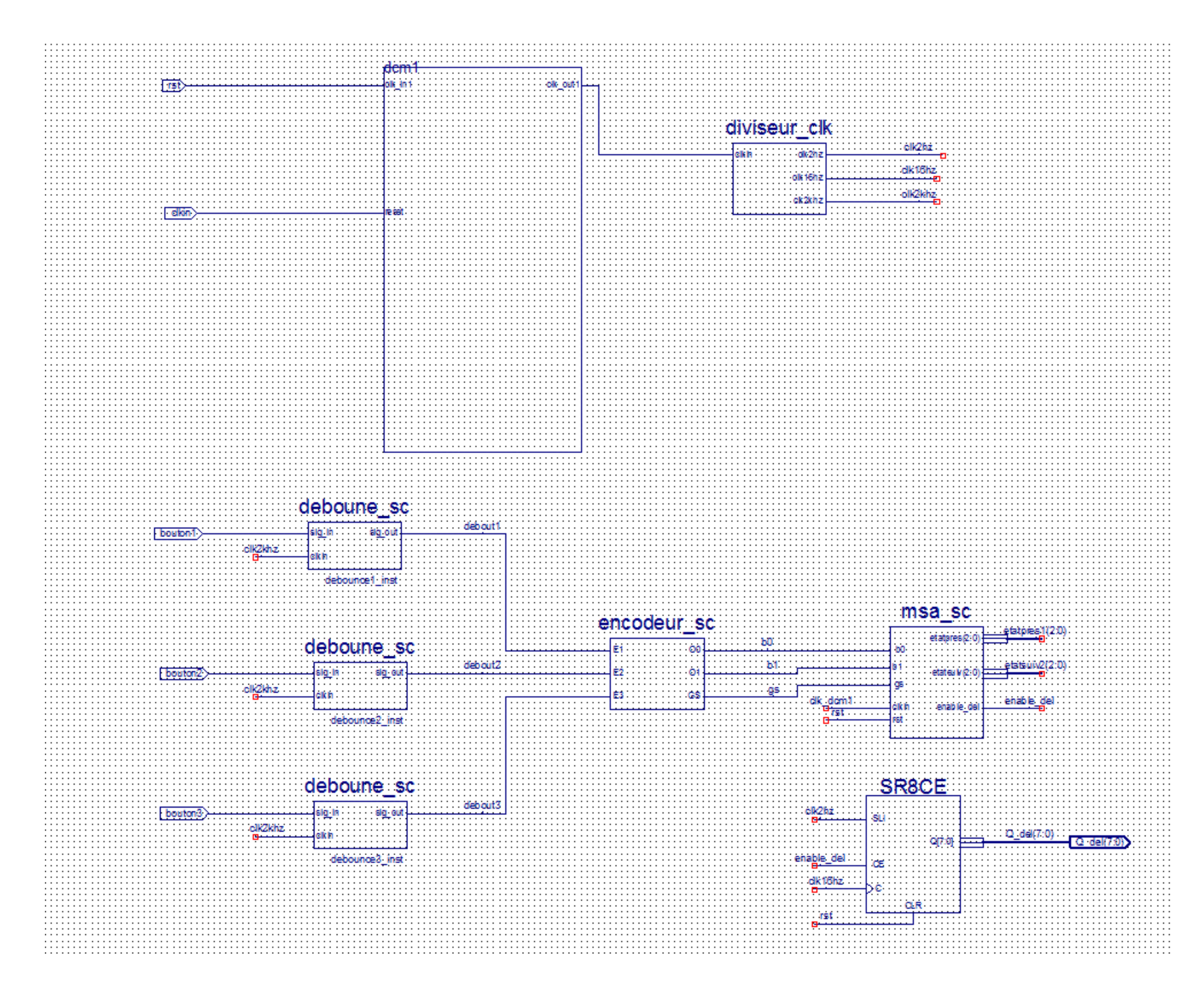



Figure 7. Schéma top level complété du détecteur de séquence synchrone.

## Création d'un nouveau projet dans ISE

Pour débuter cette partie du didacticiel, créez un nouveau projet ISE ayant pour cible le FPGA Spartan 3E.

1. Pour lancer la suite ISE version 12.4, double-cliquez sur l'icône du **Navigateur de projet ISE**, ou sélectionnez **Start > Programs > Xilinx ISE design suite 12.4 > ISE Design Tolls > 32-bit Project Navigator**. Sous Linux il faut suivre le chemin équivalent. Si votre processeur est 64 bits , utilisez la version adéquate.

Pour certaines machines, il faut supprimer la définition de la variable LANG.
Cela dépend du shell utilisé, sous Linux :
- famille csh faire 'unsetenv LANG' dans le terminal avant de lancer ise
- famille bourne shell : faire 'unset LANG'




Figure 8. Icône du Navigateur de projet ISE 12.4 disponible sur le bureau.

2. Dans le Navigateur de projet ISE, cliquez sur **File > New Project**.
3. Dans la fenêtre *New Project Wizard* , tapez **didact_sc** comme nom de projet.

Notez que didact_sc est ajouté au chemin initial contenu dans le champ Location.

4. Sélectionnez **Schematic** pour le champ Top-Level Source Type et cliquez sur **Next**.
5. Entrez les valeurs suivantes dans la fenêtre *New Project Wizard – Project Sttings*.
   - Product Category: All
   - Family: Spartan6
   - Device: XC6SLX16
   - Package: CSG324
   - Speed: -3
   - Synthesis Tool: XST (VHDL/Verilog)
   - Simulator: Isim (VHDL/Verilog)
   - Preferred Language: VHDL

La Figure 9 montre la bonne configuration à avoir pour cette boîte de dialogue.

6. Enfin, cliquez sur **Next**.

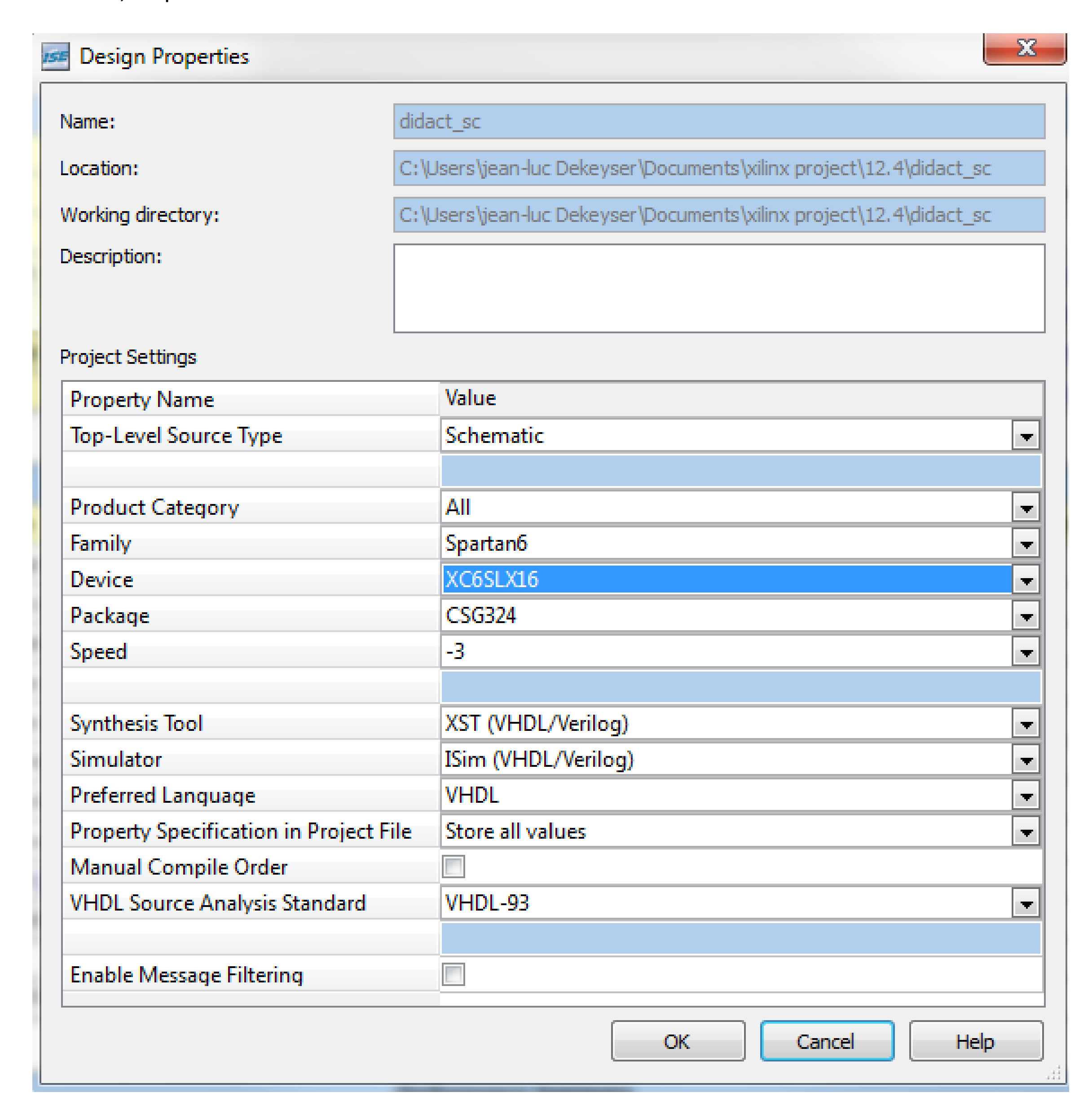



**Figure 9. Fenêtre New Projet Wizard – Project Settings: paramètres à entrer pour la configuration du FPGA Spartan 6 utilisé pour ce didacticiel.**

7. Cliquez enfin sur **Finish** dans la fenêtre *Project Summary*.

À ce stade, vous obtenez un nouveau projet ISE ne contenant encore aucun schéma, fichier HDL ou module IP.

## Création d'un schéma top level

Créez un nouveau schéma top level de la façon suivante :

1. Assurez-vous que l'onglet Design est sélectionné dans le Navigateur de Projet et cliquer sur **Project > New Source** dans la barre de menu. Start Design Files Libraries
2. Sélectionnez **Schematic** dans la fenêtre *Project Wizard – Select Source Type*.
3. Tapez **didact_top** comme nom de fichier et vérifiez que la case **Add to project** est cochée.
4. Cliquez sur **Next** et cliquez enfin sur **Finish**.

Sélectionnez l'onglet Design à nouveau et vérifiez que le champ View est bien coché sur Implementation.

Vous pouvez remarquer qu'un nouveau schéma (le schéma didact_top) est ajouté au projet en tant qu'instance top level. Le symbole correspond au top level

## Éditer le nouveau schéma didact_top.

Le schéma top level permettra d'instancier plusieurs modules de niveaux hiérarchiques inférieurs et d'interconnecter le tout pour réaliser le schéma complet du détecteur de séquence. Vous allez commencer la construction de ce schéma.

Double-cliquez sur l'item **didact_top** situé dans l'espace Hierarchy, ou alternativement, cliquez avec le bouton droit de la souris sur l'item didact_top et sélectionnez **Open**. Le fichier vide du schéma top level apparaît dans l'espace de travail. Un cadre entoure le schéma à réaliser, comme illustré à la Figure 10. L'onglet Symbols devient actif, un ensemble de symboles apparaissent dans le navigateur.

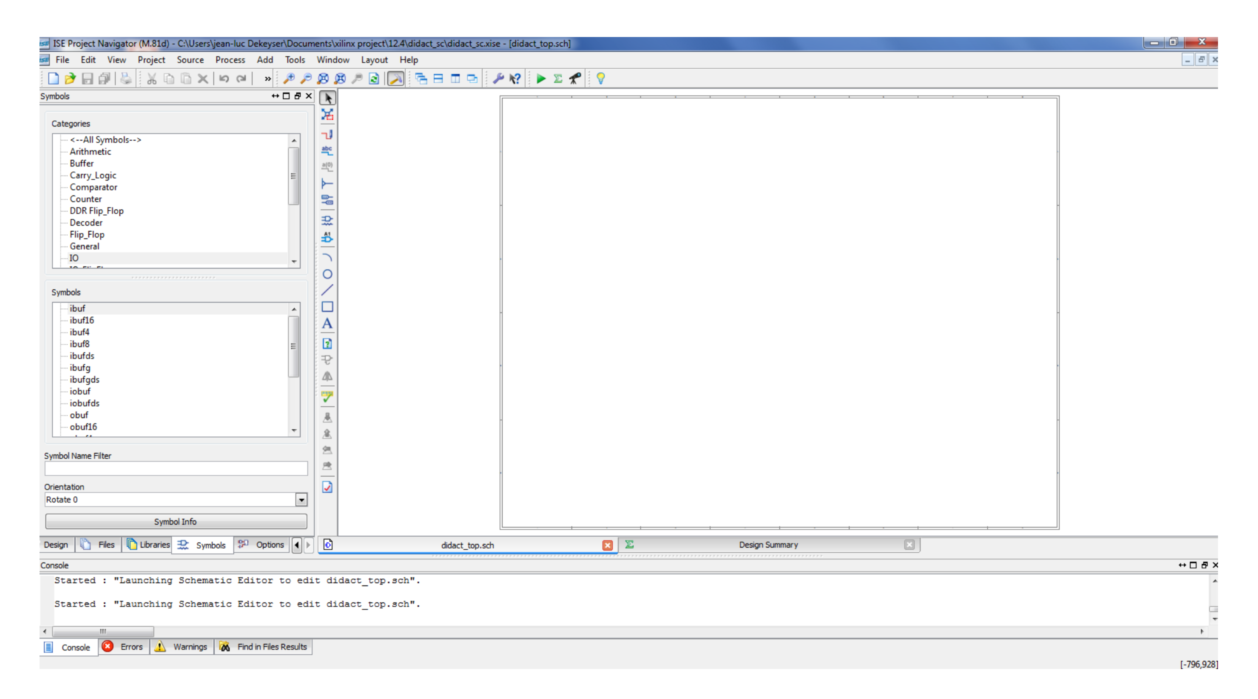


**Figure 10. Nouveau projet et schéma top level associé dans le Navigateur de projet ISE. La page correspondant au schéma didact_top est ouverte dans l'espace de travail et prête à être éditée.**

## Création d'un module schématique anti-rebond

Les boutons poussoir et les commutateurs disponibles sur la carte Spartan 6 ne disposent pas de circuits d'anti-rebond. Par conséquent, les rebonds doivent être traités à l'aide du FPGA. Pour ce didacticiel, vous allez créer un module schématique d'anti-rebond pour traiter les trois boutons utilisés pour entrer la séquence. Ce composant permet aussi de générer un signal d'implusion qui détecte le moment où le bouton est appuyé.

Les étapes suivantes montrent comment créer un module schématique d'anti-rebond à l'aide du New Source Wizard du Navigateur de projet ISE. Un module schématique est composé d'un symbole et d'un schéma qui décrit sa circuiterie et ses interconnexions. Par conséquent, vous créerez tout d'abord une page de schéma vide et définirez ensuite la circuiterie du module d'anti-rebond. Ensuite, vous créerez son symbole et le module schématique sera ajouté automatiquement à la librairie de votre projet.

Pour créer le module schématique d'anti-rebond:

1. Sélectionnez **Project > New Source.**

2. Sélectionnez **Schematic** dans la boîte de dialogue *Select Source Type* et entrez debounce_sc comme nom de fichier.
3. Vérifiez que la case **Add to project** est cochée, cliquez sur **Next** et enfin sur **Finish**. La page vide du schéma du module debounce_sc s'ouvre dans l'espace de travail.

Assurez-vous que l'onglet Design est sélectionné. Remarquez que l'item debounce_sc est ajouté au projet dans l'espace Hierarchy.

## Implémentation du schéma de l'anti-rebond

Vous avez créé une page de schéma vide pour debounce_sc. L'étape suivante consiste à ajouter les composants constituant le module d'anti-rebond et à construire le module schématique. Ce module pourra ensuite être instancié en tant que symbole dans le schéma top level et utilisé comme une macro.

### Ajoutez les étiquettes d'entrées/sorties

Les étiquettes d'entrées/sorties définissent les ports de la macro qui seront accessibles dans le schéma top level. Ces ports correspondront aux broches du symbole représentant le module.

Pour ajouter des étiquettes d'entrées/sorties :

1. Sélectionnez **Tools > Create I/O Markers.**

La boîte de dialogue Create I/O Markers s'ouvre.

2. Tapez **sig_in, clkin** dans le champ Inputs et tapez **sig_out** dans le champ Outputs.
3. Cliquez sur **Ok**. Les trois étiquettes sont ajoutées au schéma.

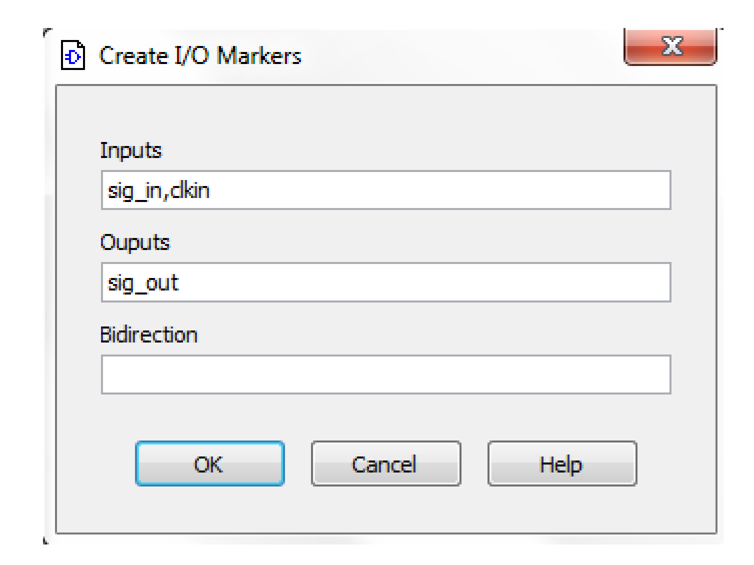



Figure 11. La boîte de dialogue Create I/O Markers.

## Construisez le circuit du module anti-rebond

Les composants disponibles dans les librairies du Spartan 6 et les autres librairies de projets peuvent être ajoutés au schéma grâce au Symbols Browser.

Pour ajouter les composants du circuit d'anti-rebond :

1. Sélectionnez **Add > Symbol** ou cliquez sur l'icône **Add symbol** de la barre d'outils ou activez l'onglet Symbols au lieu de Design.



Figure 12. Icône Add Symbol de la barre d'outils.

L'éditeur de symbole s'ouvre à gauche de l'espace de travail sous Categories, présentant les librairies de symboles et les composants qu'elles contiennent.

Vous devez tout d'abord placer trois bascules D :

1. Sélectionnez l'item **Flip_Flop** de la liste Categories.
2. Sélectionnez ensuite de type le bascule **fd** dans la liste Symbols.
3. Revenez ensuite dans l'espace de travail avec la souris. Remarquez que le curseur représente le composant à placer.
4. Alignez le fil de l'étiquette sig_in avec la broche supérieure gauche de la bascule D et cliquez avec le bouton gauche de la souris pour placer le composant.

**Note :** Pour obtenir une description détaillée d'un composant appartenant à une librairie Xilinx, cliquez sur le symbole avec le bouton droit de la souris et sélectionnez **Object Properties**. La boîte de dialogue Instance Attributes s'ouvre. Cliquez sur le bouton **Symbol Info** à droite pour accéder à la description détaillée du composant.

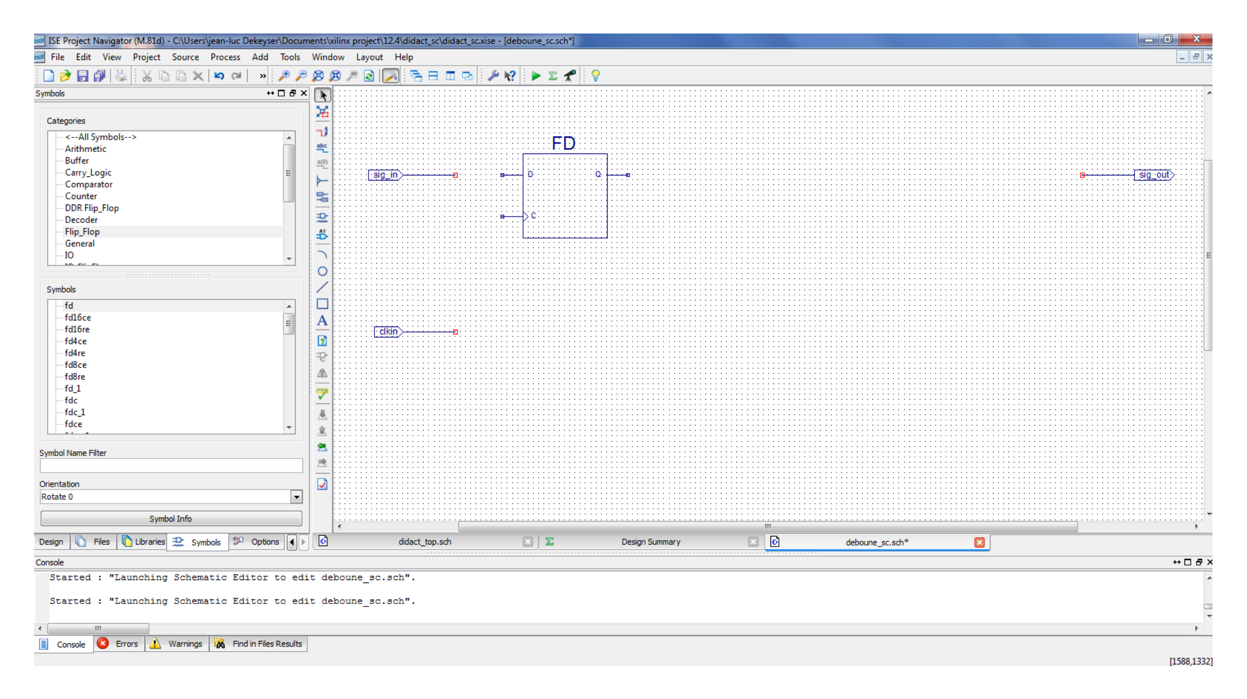



Figure 13. Le Symbol Browser permet de choisir les symboles à placer dans le schéma parmi plusieurs librairies.

5. Placez deux autres bascules D à la suite de la première. Alignez la broche D de la bascule à placer avec la broche Q de la bascule précédente. Référez-vous à la Figure 16 montrant le circuit de l'anti-rebond complété pour vous guider.

### Renommez les instances

Le Navigateur de projet ISE assigne automatiquement un nom générique à toutes les instances ajoutées à un schéma. Or, il est important d'attribuer un nom évocateur aux composants d'un schéma car cela facilite grandement les manipulations subséquentes et clarifie les résultats lors de la simulation fonctionnelle. Pour renommer une bascule D, procédez comme suit :

1. Cliquez sur la première bascule du schéma de l'anti-rebond avec le bouton de droite de la souris.
2. Sélectionnez **Object Properties**.

La boîte de dialogue *Instance Attributes* s'ouvre.

3. Changez la valeur du champ InstName pour **Q0_inst** et cliquez sur **Ok**.

Renommez les deux autres bascules D **Q1_inst** et **Q2_inst** respectivement.

### Ajoutez les autres composants du schéma

Placez ensuite un inverseur et une porte ET à la suite des bascule D, comme montré à la Figure 16. Pour ce faire :

1. Sélectionnez l'item **logic** dans la liste Categories de l'espace Symbols.
2. Sélectionnez l'item **inv** dans la liste Symbols.
3. Placer le curseur dans l'espace de travail et alignez l'entrée de l'inverseur avec la broche de sortie Q de la troisième bascule D. Cliquez pour placer le composant.

**Note :** Vous pouvez déplacer les étiquettes d'entrées/sorties pour faire de la place dans le centre du schéma et ajouter plus de symboles entre ceux-ci. Il suffit de sélectionner l'icône **Select** de la barre d'outils et de cliquez sur un composant à déplacer avec le bouton gauche de la souris en le maintenant. Glissez le curseur en le maintenant le bouton gauche enfoncé pour déplacer le composant.

Placez ensuite la porte ET en suivant les mêmes étapes 1 à 3 énoncées ci haut.

1. Sélectionnez le symbole **and3** situé dans la Catégorie **logic**.
2. Revenez dans l'espace de travail, alignez la broche inférieure de la porte ET avec la broche de sortie de l'inverseur et cliquez avec le bouton de gauche pour placer le composant.

## Interconnectez les symboles

Utilisez l'icône Add Wire de la barre d'outils pour dessiner des fils et interconnecter tous les symboles et les étiquettes du schéma.

Procédez comme suit pour interconnecter le fil relié à l'étiquette sig_in à la broche D de la première bascule :

1. Sélectionnez **Add >Wire** dans le Navigateur de projet ou cliquer sur l'icône **Add Wire** dans la barre d'outil.



Figure 14. L'icône Add Wire de la barre d'outils.

2. Cliquez sur l'extrémité droite du fil de l'étiquette **sig_in**. Un fil s'étire de ce point jusqu'au curseur. Cliquez sur la broche D de la première bascule D pour créer une connexion.
3. Réalisez toutes les connexions du schéma conformément au schéma final du module anti-rebond montré à la Figure 16. N'hésitez pas à déplacer les composants et les étiquettes au besoin.

**Note :** Pour supprimer une connexion, il suffit de sélectionner le fil et appuyer sur la touche **Supprimer** du clavier. De plus, vous pouvez sortir du mode Add Wire en tout temps en appuyant sur la touche **Esc** ou click gauche.

Enfin, pour créer un embranchement à partir d'un fil existant, procédez comme suit :

1. Sélectionnez le mode **Add Wire** de la façon explicitée précédemment.
2. Cliquez sur le fil au point où vous souhaitez réaliser l'embranchement.
3. Terminez la connexion à l'endroit désiré en pointant la destination avec pointeur et en cliquant avec le bouton gauche de la souris.

## Ajouter les noms de signaux

Il est très important d'identifier les principaux signaux d'un schéma à l'aide d'un nom approprié. Les fils prennent le nom de l'étiquette à laquelle ils sont reliés automatiquement. Pour nommer les autres fils du schéma effectuez ces étapes :

1. Sélectionnez **Add > Net Name** ou cliquez sur l'icône **Add Net Name** de la barre d'outils.



Figure 15. L'icône Add Net Name de la Barre d'outils.

2. Dans la l'espace Options (à gauche, sous l'espace Symbols), tapez **Q0** dans le champ Name.
3. Sélectionnez l'option **Increase the name** dans le bloc After naming the branch or net.
4. Cliquez successivement sur les trois tronçons de fils connectés à la broche Q de chacune des trois bascules D pour identifier chaque fil.
5. Cliquez sur **File > Save**.

Le schéma du circuit anti-rebond est maintenant complété. Le résultat obtenu devrait être conforme au schéma montré à la Figure 16.

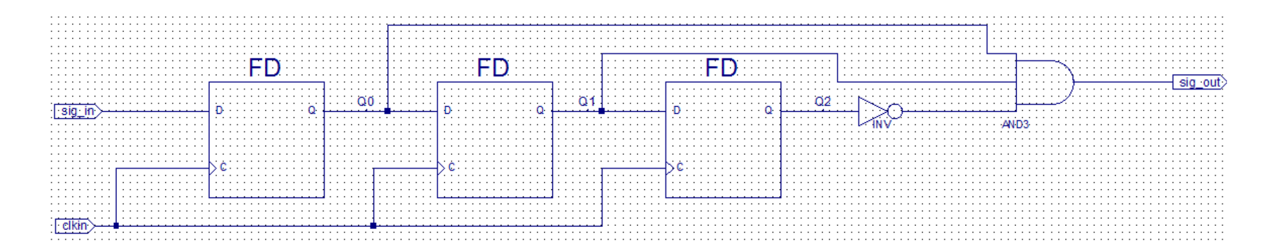



Figure 16. Le schéma du circuit d'anti-rebond complété.

## Vérification du schéma

Maintenant que le schéma est complété, effectuez un design rule check pour vous assurer que qu'il ne contient aucune erreur. Pour ce faire :

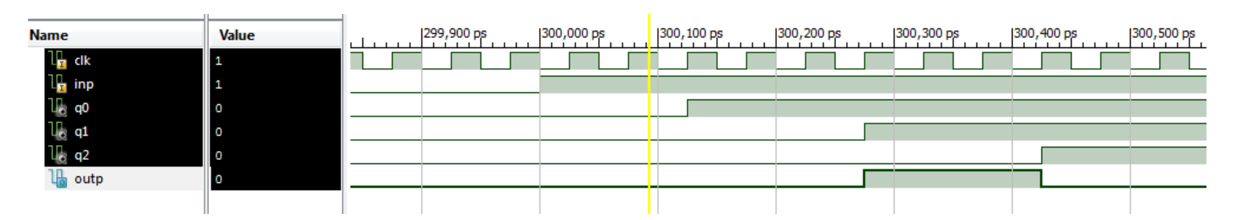
1. Sélectionnez **Tools > Check Schematic.** Le résultat du DRC s'affiche dans la Console (partie située dans la partie inférieure du Navigateur de projet). Corrigez les erreurs et éliminez les avertissements de type warnings s'il y a lieu.
   On peut aussi passer par l'onglet design en sélectionnant le fichier deboune_sc en mode implementation, il suffit alors de lancer Check Design Rules dans Design Utilities
2. Fermer le schéma du module anti-rebond.

## Création du symbole du module anti-rebond

Les prochaines étapes consistent à créer le symbole du module anti-rebond et à le placer dans le schéma top level.

1. Sélectionnez l'onglet **Design** à gauche du Navigateur de projet et sélectionnez l'item **debounce_sc** de l'arborescence du projet, dans l'espace Hierarchy.
2. Déployer la liste d'item **Design Utilities** en cliquant sur le symbole **+** lui faisant face.
3. Double-cliquez sur l'item **Create Schematic Symbol**.

Alternativement, vous pouvez créer un symbole en sélectionnant **Tools > Symbol Wizard** lorsque le schéma à représenter est ouvert dans l'espace de travail. Ce composant pourrait être vérifié par simulation. Voici ile résultat que vous devriez obtenir.




## Placez le symbole debounce_sc

Vous pouvez maintenant placer le symbole du module anti-rebond dans le schéma top level. Procédez comme suit :

1. Double-cliquez sur l'item **didact_top** dans l'arborescence du projet.

Le schéma top level s'ouvre dans l'espace de travail et l'onglet Symbols est du même coup sélectionné, montrant les librairies et composants disponibles.

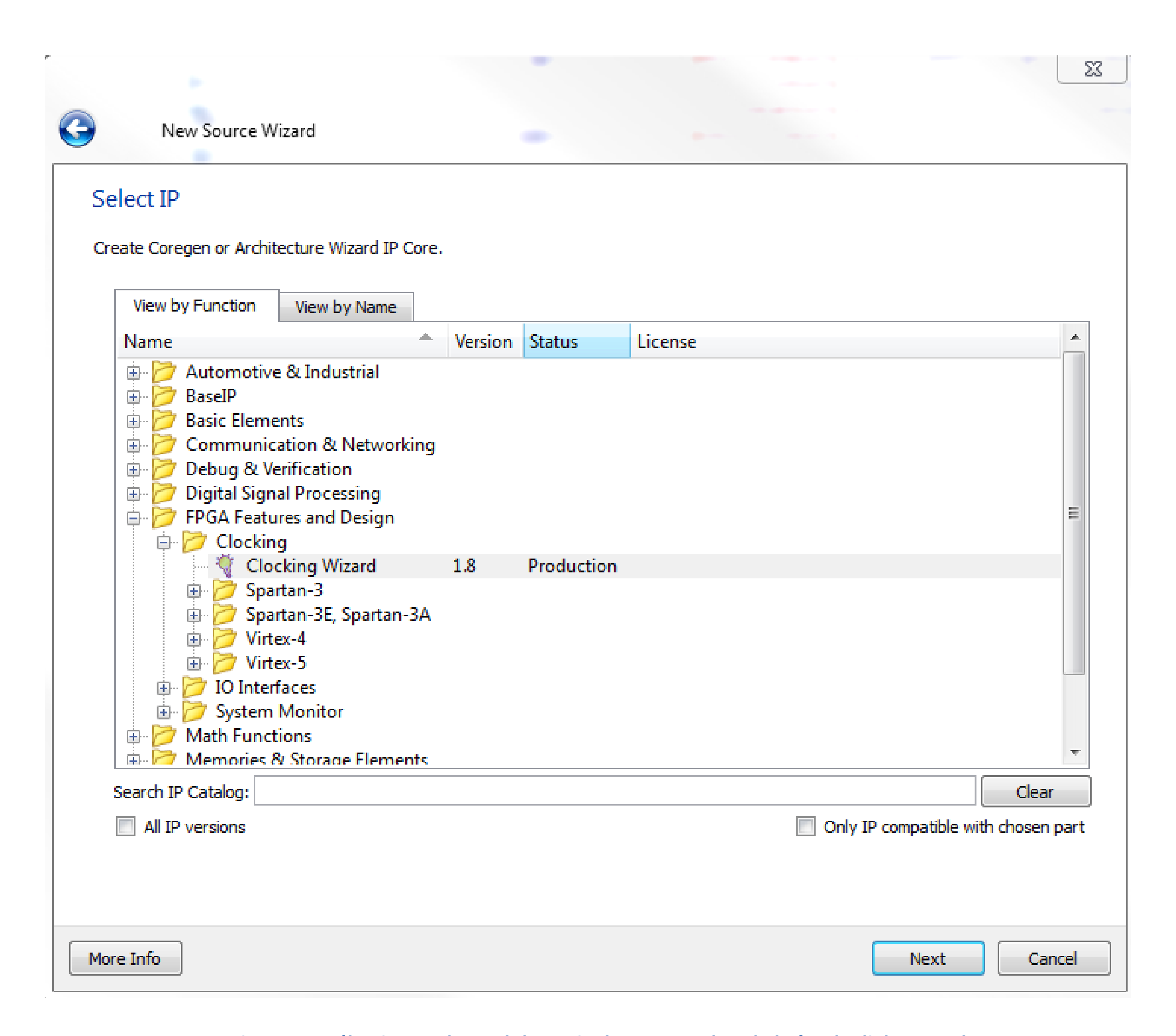
2. Sélectionnez **Add > Symbol** ou cliquez sur l'icône **Add symbol** de la barre d'outils ou l'onglet Symbols.
3. Dans la liste Categories, sélectionnez la librairie de symboles locale ***« work_space_path »*/didact_sc**.
4. Sélectionnez le symbole **debounce_sc** dans la liste Symbols et placez-en trois instances dans la partie gauche de la page, comme montré à la Figure 7 présentant le schéma top level complété.
5. Renommez les trois instances du module debounce_sc en tant que **debounce1_inst**, **debounce2_inst** et **debounce3_inst** dans l'ordre en partant du haut. On peut rendre le nom visible en cochant la case *Visible*.
6. Sauvegardez le schéma top level en sélectionnant **Save All**.

Remarquez que les trois instances debounce_sc ajoutées au schéma top level apparaissent sous l'item didact_top dans l'arborescence du projet.

## Création d'un module de gestion d'horloge

L'Architecture Wizard de ISE permet de créer et de configurer graphiquement une panoplie de module IP (Intellectual Properties) très rapidement. Vous allez créer et configurer un module de gestion d'horloge (DCM) avec feedback et correction automatique de rapport cyclique grâce à cet outil. Le DCM dérivera une horloge d'une fréquence de 16 MHz à partir de l'horloge de référence de 100 MHz disponible sur la carte Spartan 6.

1. Cliquer sur **Project > New Source** dans la barre de menu du Navigateur de Projet ISE.
2. Sélectionnez **IP (CORE Generator & Architecture Wizard)** dans la fenêtre *Project Wizard – Select Source Type*.
3. Tapez **dcm1** comme nom de fichier, vérifiez que la case **Add to project** est cochée et cliquez sur **Next**.
4. Sélectionnez **FPGA Features and Design > Clocking > Clocking Wizard** dans l'onglet View by fonction de la boîte de dialogue Select IP, comme illustré à la Figure 17.
5. Cliquez sur **Next** et ensuite sur **Finish**.




6. Figure 17. Sélectionnez le module IP Single DCM_IP dans la boîte de dialogue Select IP.

ISE lance alors le *Xilinx Clocking Wizard – General Setup*, illustré à la Figure 18. Dans cette boîte de dialogue tapez **Next**. Vous arrivez sur la deuxième fenêtre :

1. Dans la pageOutput Frequency Requested de CLK-OUT1, entrez **16**.
2. Cliquez sur **Next**.
3. Sur la page 3 il faut désélectionner **LOCKED**.
4. Sur la page 6 vous obtenez la liste des fichiers générés. Pour cela il suffit de cliquer sur **Generate**.

Le module dcm1 (fichier dcm1.xco) est ajouté dans l'arborescence du projet de l'espace Hierarchy.

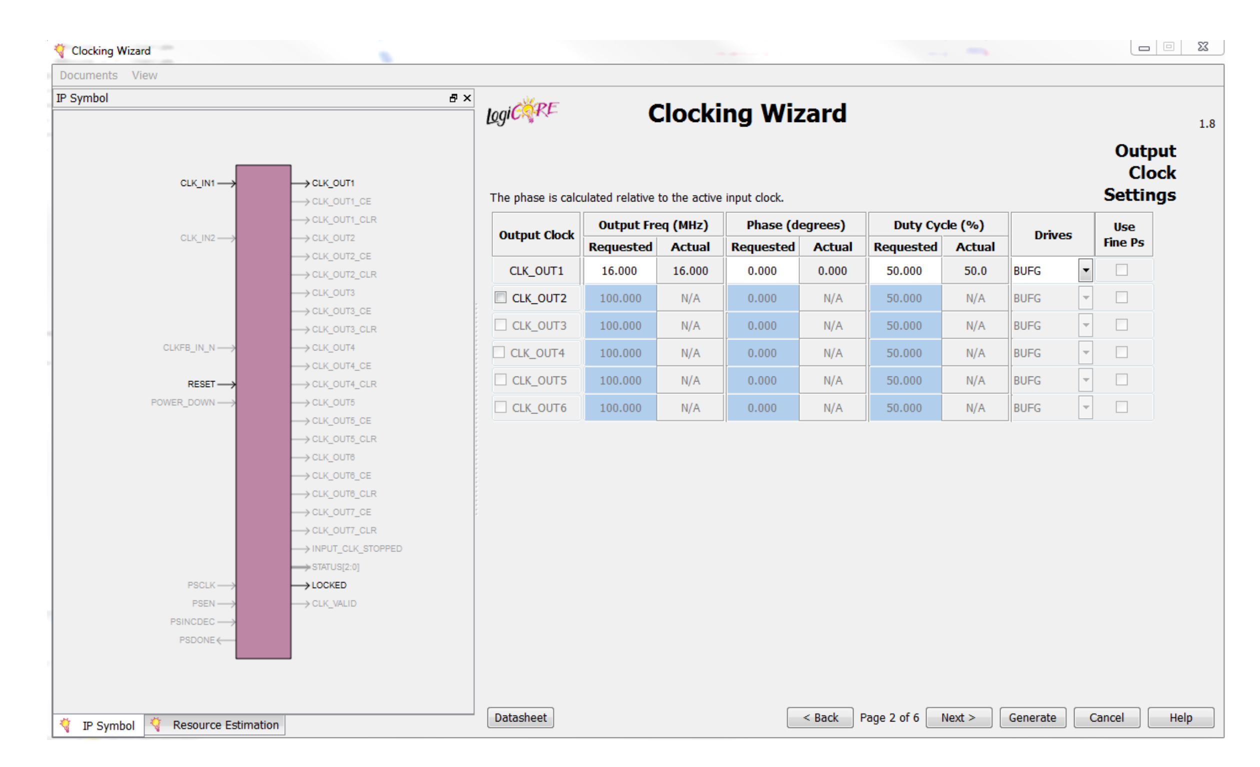



Figure 18. Le Clocking Wizard de ISE permet de générer et configurer des modules DCM sur mesure.

Le symbole représentant le module dcm1 est créé par ISE et placé dans la librairie locale de projet …./didact_sc/ipcore_dir.

## Création d'un module VHDL diviseur d'horloge

Il est facile de créer des modules décris en langage VHDL dans ISE. Une fois créés, ces modules peuvent être représentés par des symboles et interconnectés avec d'autres instances dans un schéma. Vous allez créer un diviseur d'horloge décrit en langage VHDL.

Pour créer le fichier source du diviseur :

1. Sélectionnez **Project > New Source.**
2. Sélectionnez **VHDL Module** dans la boîte de dialogue Select Source Type.
3. Entrez **diviseur_clk** comme nom de fichier et appuyer sur **Next**.

4. Créez un port d'entrée **clkin** et trois ports de sortie **clk2hz, clk16hz** et **clk2khz** comme illustré à la Figure 19.
    - Entrez **clkin, clk2hz**, **clk16hz** et **clk2khz** dans les 4 premiers champs Port Name.
    - Dans le champ Direction, choisissez **in** pour clkin, et **out** pour clk2hz, clk16hz et clk2khz.
    - Ne cochez pas les champs Bus.

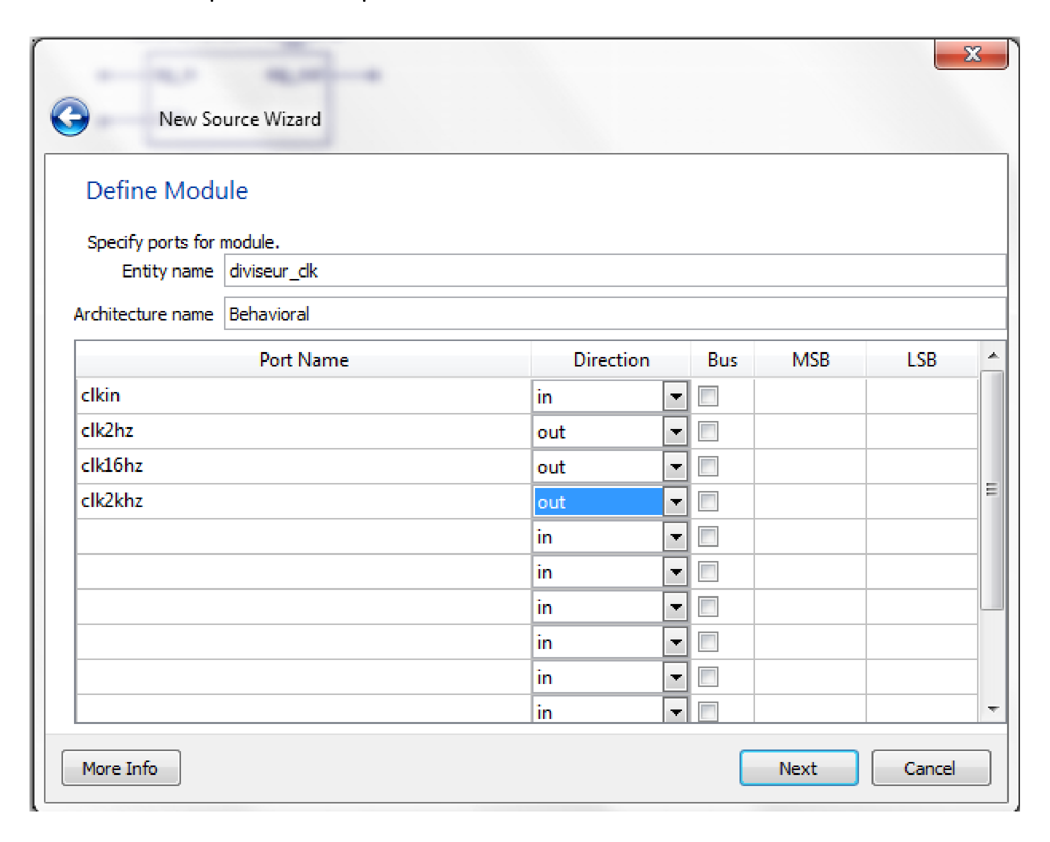



**Figure 19. La boîte de dialogue Define Module pour entrer les ports d'entrées/sorties du module VHDL à créer.**

5. Cliquez sur **Next** et ensuite sur **Finish** pour compléter cette étape de configuration.

Une boîte de dialogue affiche la description VHDL générée par le Wizard.

6. Cliquez sur **Finish**.

Le fichier VHDL vide décrivant le module diviseur_clk s'ouvre dans l'espace de travail comme montré à la Figure 20. Ce fichier contient l'entité du module mais présente une architecture vide à compléter.

## Complétez la description VHDL de diviseur_clk

Pour compléter la description VHDL du module diviseur_clk, effectuer les étapes suivantes :

1. Dans le fichier diviseur_clk ouvert dans l'espace de travail du Navigateur de Projet ISE, effacez **toutes** les lignes de code situées sous la ligne **`architecture Behavioral of diviseur_clk is.`**
2. Aller à l'Annexe 1 de ce document et visualisez la description VHDL originale du module HDL diviseur_clk. À l'aide de la souris, sélectionnez **toutes** les lignes de code situées sous la ligne **`architecture Behavioral of diviseur_clk is`** de cette description.
3. Copiez les lignes de code sélectionnées et collez-les dans votre fichier diviseur_clk dans ISE, sous la ligne **`Behavioral of diviseur_clk is`**.
4. Faire de même pour les commandes **use** en début de programme.

Vérifiez le tout au besoin pour éviter de provoquer des erreurs de syntaxe lors de la compilation du fichier.

Le code VHDL du module du diviseur d'horloge est maintenant complété.

5. Cliquez sur **File > Save**.
6. Sélectionnez l'item **diviseur_clk** dans l'espace Hierarchy et double-cliquez sur **Check Syntax** dans l'espace Processes.
7. Vérifiez les résultats dans la Console et corrigez les erreurs de syntaxe s'il y a lieu. Fermer ensuite le fichier diviseur_clk.vhd.

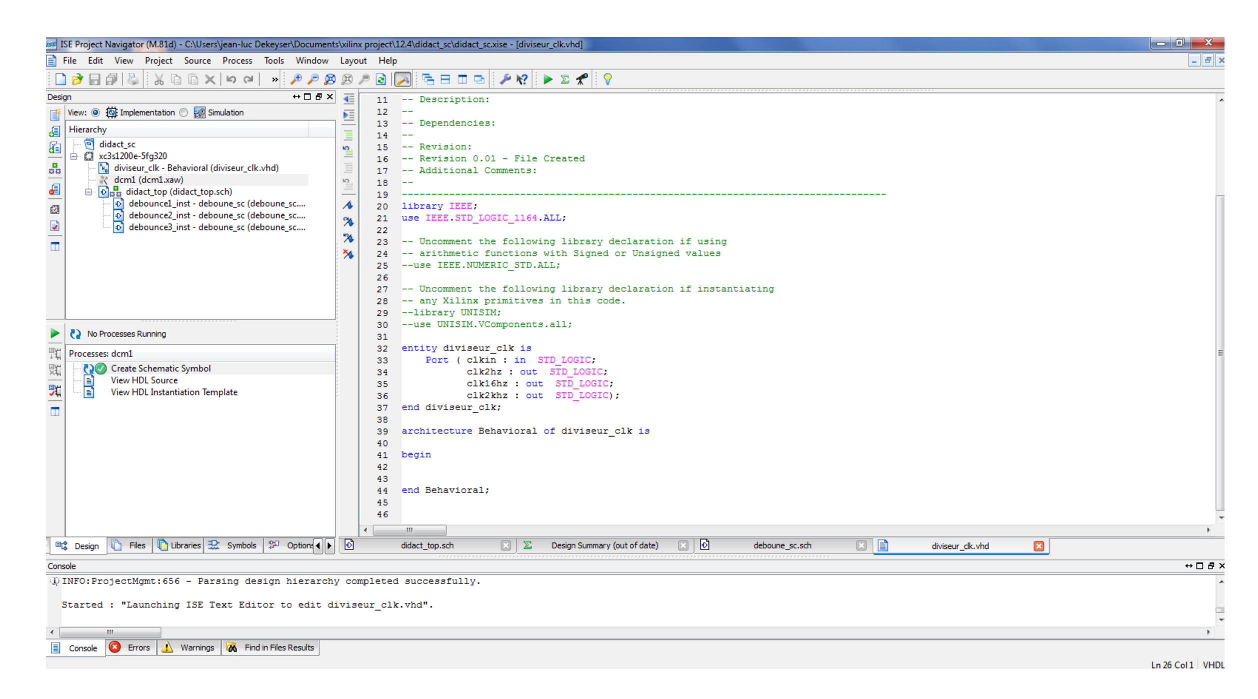


Figure 20. Fichier VHDL incomplet décrivant le module diviseur_clk.

## Création du symbole du diviseur d'horloge

Ensuite, créez un symbole représentant le module VHDL diviseur_clk dans le Navigateur de Projet. Ce symbole sera ajouté dans le schéma top level (didact_top.sch) plus tard dans le didacticiel.

1. Sélectionnez l'item **diviseur_clk** dans l'espace Hierarchy du Navigateur de Projet.
2. Dans l'espace Processes, cliquez sur le **+** en avant de Design Utilities pour déployer l'arborescence associée.
3. Double-cliquez sur **Create Schematic Symbol**.

Le symbole du diviseur d'horloge est prêt à être placé dans le schéma top level didact_top.

## Placez les symboles des modules dcm1 et diviseur_clk

Vous allez placer les symboles des modules dcm1 et diviseur_clk dans le schéma top level. Suivez les étapes suivantes :

1. Dans l'espace Hierarchy du Navigateur de projet, double-cliquez sur l'item **didact_top**. Le fichier du schéma top level s'ouvre dans l'espace de travail.

2. Sélectionnez **Add > Symbol** ou cliquez sur l'icône **Add symbol** de la barre d'outils ou onglet Symbols.



Figure 21. Icône Add Symbol de la barre d'outils.

L'éditeur de symbole s'ouvre à gauche de l'espace de travail. L'éditeur de symbole présente les librairies de symboles (sous Categories) et les composants qu'elles contiennent (sous Symbols).

3. Repérez les deux librairies correspondant au projet en cours dans l'espace Categories et sélectionnez-les une à une.
4. Sélectionnez le symbole **dcm1** dans l'espace Symbols sous Categories et placez-le dans la partie supérieure gauche de la page du schéma didact_top, ouvert dans l'espace de travail.
5. Faite de même pour le symbole diviseur_clk dans la seconde librairie. Laissez les modules tel quel sans les connecter pour le moment.
6. Renommez l'instance du module dcm1 **dcm1_inst** et l'instance du diviseur **diviseur_inst** en appliquant l'approche détaillée précédemment à la section Renommez les instances.
7. Sauvegardez le schéma top level et fermez-le.

## Création d'un module schématique encodeur

Il peut être très utile d'encoder plusieurs signaux pour les représenter à l'aide d'un nombre moindre de bit. Vous allez créer un encodeur pour représenter les signaux provenant des trois boutons (BTNL, BTNS et BTNR) sur 2 bits. Procédez comme suit :

1. Tout d'abord, créez un nouveau module schématique nommé **encodeur_sc** à l'aide du New Source Wizard. Reprenez les étapes vues précédemment aux sections Création d'un module

schématique anti-rebond et Implémentation du schéma de l'anti-rebond pour créer ce nouveau schéma.

Une fois toutes les étapes effectuées, le schéma du nouveau module **encodeur_sc** s'ouvre dans l'espace de travail. Ajouter des étiquettes d'entrées/sorties au schéma :

2. Sélectionnez **Tools > Create I/O Markers** et procédez comme fait à la section Ajoutez les étiquettes d'entrées/sorties pour ajouter des étiquettes.
3. Ajoutez les trois étiquettes d'entrées **E1, E2, E3**.
4. Ajoutez les trois étiquettes de sorties **O0, O1** et **GS**.
5. Cliquez sur **Ok** pour créer les étiquettes.

### Ajouter deux LUT au schéma

L'encodeur sera réalisé à l'aide des tables de référence (Look-up-table – LUT) disponibles dans le FPGA Spartan 6. Ces LUT sont très efficaces car elles permettent de réaliser n'importe quelle expression logique d'un nombre prédéterminé de variables lorsqu'elles sont programmées convenablement.

Les **lut3** sont des tables de référence comportant 3 variables d'entrée et une sortie. Elles se trouvent dans la Librairie LUT disponible dans l'espace Categories de l'éditeur de symbole. Effectuez ces étapes :

1. Ajoutez deux symboles **lut3** au schéma de l'encodeur à l'aide de l'éditeur de symbole. Référez-vous à la section Construisez le circuit du module anti-rebond vue précédemment pour vous guider dans cette tâche.

**Note :** Consultez la fonction **Symbol Info** pour une description détaillée de cette LUT et les explications sur la les façons de la programmer. Consultez la section Construisez le circuit du module anti-rebond pour plus d'information sur cette fonction.

2. Ajoutez une porte OU à 3 entrées au schéma. Le symbole de cette porte se trouve dans la librairie **logic** de la liste Categories.
3. Sélectionnez l'item **or3** de la liste Symbols et placez-le sous les LUT. Référez-vous au schéma complété de l'encodeur présenté à la Figure 22 pour voir la disposition suggérée.

## Interconnectez les LUT

Ensuite, réalisez les interconnexions conformément à la Figure 22. Pour ce faire, effectuez les étapes suivantes :

1. Connectez les étiquettes **E1, E2 et E3** aux broches **I0, I1 et I2** respectivement des deux LUT selon les étapes effectuées précédemment pour le module d'anti-rebond. Référez-vous à la Figure 22 pour une représentation du schéma complété.
2. Connectez les étiquettes **E1, E2 et E3** aux trois broches d'entrées respectivement de la porte OU.
3. Connectez la broche de sortie de la première LUT à l'étiquette **O0**.
4. Connectez la broche de sortie de la deuxième LUT à l'étiquette **O1**.
5. Connecter la broche de sortie de la porte OR à l'étiquette **GS**.

## Renommez les LUT

Renommez ensuite les instances avec des noms évocateurs :

1. Double-cliquez sur la première LUT. La boîte de dialogue Instance Attributes s'ouvre.
2. Inscrivez **LUT1_inst** dans le champ IntName et cliquez sur **Ok**.
3. Répétez ces deux étapes pour renommer la deuxième LUT **LUT2_inst**.

Enfin, sauvegardez le schéma.

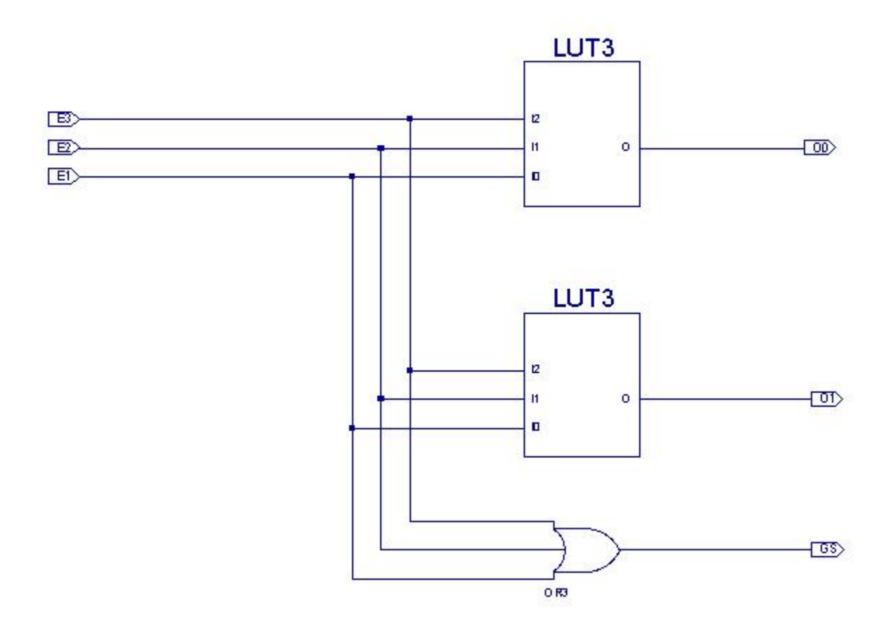



Figure 22. Schéma complété du module schématique encodeur_sc.

## Initialisez les LUT

Les deux LUT doivent être initialisées avec les chaînes de caractères appropriées pour réaliser l’expression logique souhaitée correctement.

La table de vérité correspondant à l’expression logique à réaliser pour encoder les signaux provenant des boutons poussoirs est la suivante :

| Entrées | | | Sorties | | | | Chaîne INIT |
|---|---|---|---|---|---|---|---|
| I2 | I1 | I0 | O1 | O0 | GS | | |
| 0 | 0 | 0 | 0 | 0 | 0 | → | b0 |
| 0 | 0 | 1 | 0 | 0 | 1 | → | b1 |
| 0 | 1 | 0 | 0 | 1 | 1 | → | b2 |
| 0 | 1 | 1 | 0 | 0 | 1 | → | b3 |
| 1 | 0 | 0 | 1 | 0 | 1 | → | b4 |
| 1 | 0 | 1 | 0 | 0 | 1 | → | b5 |
| 1 | 1 | 0 | 0 | 1 | 1 | → | b6 |
| 1 | 1 | 1 | 0 | 0 | 1 | → | b7 |

**Figure 23. Table de vérité de l'encodeur et chaîne d'initialisation des LUT correspondantes**

La première LUT doit réaliser la séquence associée à la sortie O0 tandis que la deuxième LUT doit réaliser la séquence associée à la sortie O1. La colonne Chaîne INIT donne les bits correspondants de la chaîne et leur position. Ainsi, la séquence à entrer pour la 1ère LUT est la suivante : 01000100 (du MSB au LSB) , alors que la séquence 00010000 (du MSB au LSB) est utilisée pour initialiser la 2ième LUT.

Par ailleurs, le format d'initialisation du composant exige que la séquence soit entrée en base hexadécimale. Par conséquent les séquences d'initialisation seront les suivantes : **44** pour la 1ère LUT et **10** pour la 2ème LUT.

Toujours avec le schéma de l'encodeur ouvert, procédez comme suit pour initialiser les LUT :

1. Double-cliquez sur la première LUT.

La boîte de dialogue Instance Attributes s'ouvre.

2. Inscrivez le nombre hexadécimal **44** comme valeur dans le champ INIT.
3. Double-cliquez sur la deuxième LUT.
4. Inscrivez le nombre hexadécimal **10** comme valeur dans le champ INIT.

**Note :** Consultez la documentation de la LUT lut3 grâce à l'aide de la fonction Symbol Info disponible dans le menu Object Properties pour en savoir davantage sur la méthode d'initialisation.

### Effectuez une vérification du schéma

Procédez comme vous l'avez fait précédemment avec le schéma du module debounce_sc :

1. sélectionnez **Tools > Check Schematic.**

Le résultat du DRC s'affiche dans la Console (partie située dans la partie inférieure du Navigateur de projet). Corrigez les erreurs et élimiez les avertissements de type warnings s'il y a lieu.

2. Fermer le schéma du module encodeur_sc.

### Créez le symbole du module encodeur_sc

Créez un symbole pour l'encodeur et placez-le dans le schéma top level. Procédez comme vous l'avez fait pour le module debounce_sc:

1. Sélectionnez l'onglet **Design** à gauche du Navigateur de projet et sélectionnez l'item **encodeur_sc** de l'arborescence du projet, dans l'espace Hierarchy.
2. Déployer la liste d'item Design Utilities en cliquant sur le symbole **+** lui faisant face.
3. Double-cliquez sur l'item **Create Schematic Symbol**.

Le symbole encodeur_sc est ajouté à la librairie locale de projet …/didact_sc.

### Placez le symbole encodeur_sc

Placer le symbole de l'encodeur dans le schéma top-level en procédant comme suit :

1. Double-cliquez sur l'item **didact_top** dans l'arborescence du projet.
2. Sélectionnez **Add > Symbol** ou cliquez sur l'icône **Add symbol** de la barre d'outils.
3. Dans la liste Categories, sélectionnez la librairie de symboles locale **U:/didact_sc**.
4. Sélectionnez le symbole **encodeur_sc** dans la liste Symbols et placez-le au centre du schéma top level, comme illustré sur le schéma complété à la Figure 7.
5. Renommez l'instance du module en tant que **encodeur_inst**.
6. Sauvegardez le schéma top level et fermez-le

## Création d'une MSA à l'aide d'un module schématique

Vous allez implémenter une MSA à l'aide d'un module schématique. Ensuite, vous allez créer un symbole pour la représenter et la connecter dans le schéma top level. Cette MSA utilise la même structure que celle représentée par le schéma de la Figure 6. Des bascules D seront utilisées pour confectionner un registre d'état alors qu'une mémoire ROM sera utilisée pour réaliser les circuits IFL et OFL.

Comme vous le verrez, il est aisé de réaliser des MSA de faibles et moyennes complexités avec une ROM. En effet, la table d'entrée/sorties et d'état futures de la MSA est pratiquement utilisée telle quelle pour programmer la ROM avec cette méthode. La table de vérité de la ROM réalisant l'IFL et l'OFL de la MSA est présentée à l'Annexe 2.

Le principe est le suivant : la valeur qui adresse la ROM correspond aux bits d'état présents et aux bits des entrées de la table, soit Adresse = [états présents, entrées] du MSB au LSB, alors que la valeur contenue à l'espace mémoire correspond aux bits d'état futurs et aux bits de sorties de la table, soit Contenu = [états futurs, sorties] du MSB au LSB.

### Créez un nouveau schéma pour la MSA

1. Tout d'abord, créez un nouveau module schématique nommé **msa_sc** à l'aide du *New Source Wizard*, comme vous l'avez fait précédemment pour les modules debounce_sc et encodeur_sc.
2. Ajustez la taille de la page du schéma :
   - Cliquer n'importe où dans la page du schéma avec le bouton droit de la souris et sélectionnez **Object Properties**.
   - Sélectionnez l'item **Sheets** de la liste Category et choisissez la valeur **D = 34 x 22 in** pour le champ Size.
   - Cliquez ensuite sur **Ok**.
3. Ajouter des étiquettes d'entrées/sorties à l'aide de la fonction **Create I/O Markers**. Créez les étiquettes suivantes (référez-vous aux sections précédentes pour plus de détails sur la façon de procéder avec les étiquettes):
   - Entrées : b0, b1, gs, clkin, rst.
   - Sorties : etatpres(2:0), etatsuiv(2:0), enable_del.

Après cette étape, les étiquettes b0, b1, gs, clkin et rst se retrouve à gauche du schéma alors que etatpres(2:0), etatsuiv(2:0) et enable_del se retrouvent à droite.

## Création d'une mémoire ROM à l'aide de CORE Generator

CORE Generator est un outil avec interface graphique permettant de créer des modules de complexité supérieure dont des mémoires, des fonctions mathématiques, des interfaces de communications, etc. CORE Generator permet d'optimiser les modules créés pour exploiter les caractéristiques particulières à chaque FPGA Xilinx. Vous allez utiliser CORE Generator pour générer une ROM comportant 48 éléments d'une taille de 4 bits. Cette ROM réalisera le circuit IFL et le circuit OFL de la MSA.

Suivez les étapes suivantes pour créer une ROM à l'aide de CORE Generator :

1. Cliquer sur **Project > New Source** dans la barre de menu du Navigateur de Projet ISE.
2. Sélectionnez **IP (Coregen & Architecture Wizard)** dans la fenêtre Project Wizard – Select Source Type.
3. Tapez **rom_msa** comme nom de fichier, vérifiez que la case Add to project est cochée et cliquez sur **Next**.
4. Dans l'onglet View by fonction de la boîte de dialogue Select IP, sélectionnez **Memories & Storage Elements > RAMs & ROMs > Distributed Memory Generator.**
5. Cliquez sur **Next** et ensuite sur **Finish**.

ISE lance l'interface de génération et configuration de mémoire ROM (Distributed memory generator) illustré à la Figure 24.

6. Remplissez les champs du Distributed memory generator de la façon suivante (voir le résultat à la Figure 24) :
    - Depth : **48**
    - Data Width : **4**
    - Memory Type : **ROM**
    - Cliquez sur **Next.**
    - Assurez-vous que les Input Options et les Output Options sont à **Non Registered** et cliquez sur **Next.**

**Note :** Cliquez sur le bouton **Data sheet** pour accéder au manuel du Distributed Memory Generator et pour en savoir plus sur ses fonctions ainsi que sur les caractéristiques des mémoires disponibles et sur les façons de les initialiser.

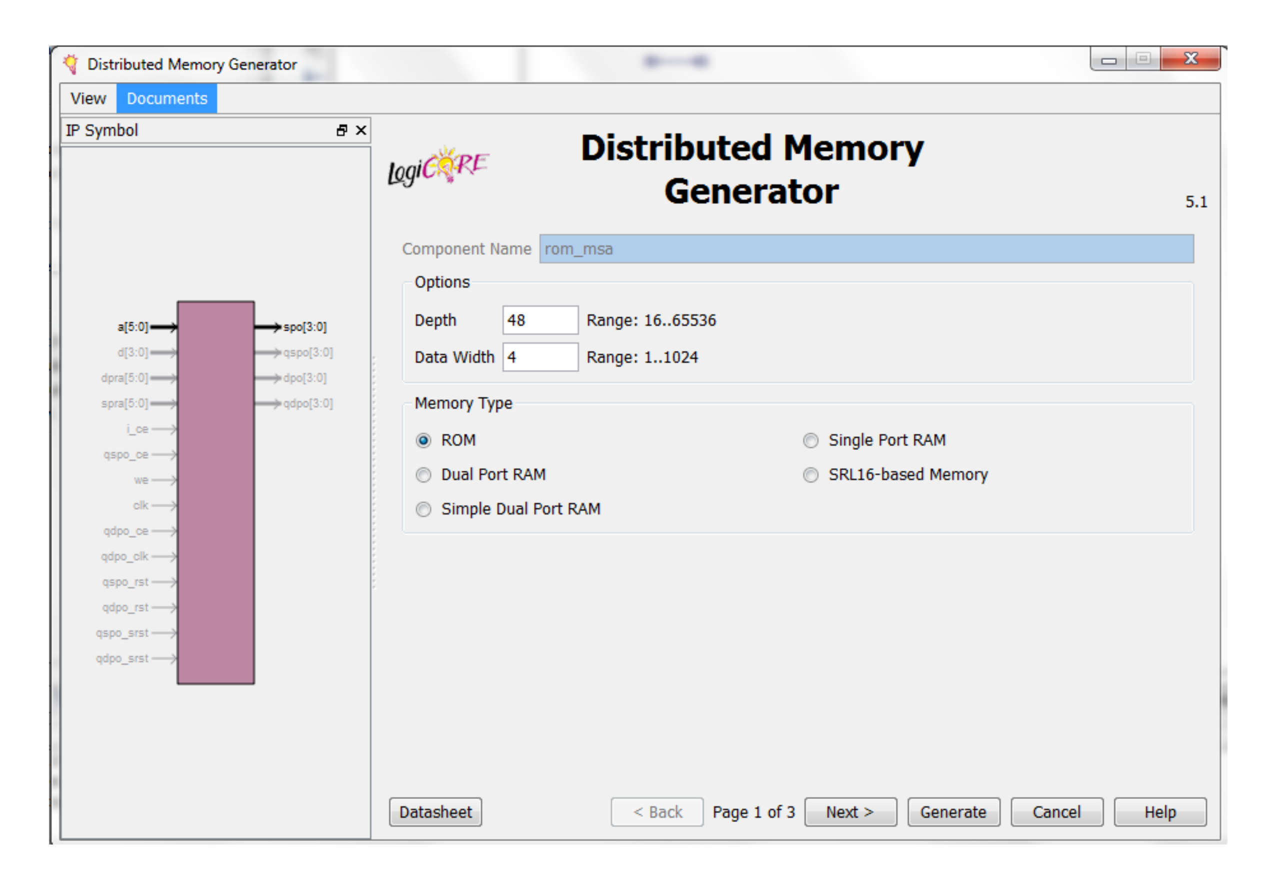



**Figure 24. Le Distributed memory generator de ISE permet de créer et de configurer n'importe quel type de mémoire très efficacement.**

Ensuite, vous devez créer un fichier d'initialisation pour que le module rom_msa réalise la table d'états futurs souhaités. Ce fichier servira à initialiser la valeur des espaces mémoires correspondant aux adresses de la mémoire ROM. Un fichier d'initialisation a été placé à l'Annexe 3 à ces fins. Pour ce faire :

1. Copiez le contenu du fichier d'initialisation du module rom_msa présenté à l'Annexe 3.
    - Sélectionnez toute les lignes du fichier à l'aide de la souris.
    - Copiez les lignes du fichier.
2. Retournez dans le Distributed Memory Generator et cliquez sur **Browse** à côté du champ Coefficients File.

La boîte de dialogue Select COE File s'ouvre.

3. Sous windows, dans la liste Files of type située dans le bas de la boîte de dialogue, Sélectionnez **All Files (*)**.
4. Cliquez dans le centre de la boîte de dialogue (dans l'espace blanc à côté des répertoires et fichiers existants) avec le bouton droit de la souris et sélectionnez **New > Text Document**.
5. Le fichier New Text Document.txt est créé.
6. Changez le nom de ce fichier pour **rom_msa.coe.**
7. Cliquez sur ce nouveau fichier avec le bouton droit de la souris et sélectionnez **Open** dans le menu déroulant.
8. Windows vous suggère une liste d'outils avec lesquels ouvrir ce fichier.
9. Choisissez d'ouvrir le fichier avec **Notepad**.
10. Sous linux faites la même chose pour créer le fichier…

Un fichier vide intitulé rom_msa.coe s'ouvre dans Notepad.

11. Collez le contenu du presse papier (fichier original copié à l'Annexe 3) dans ce fichier vide.
12. Sauvegardez le fichier **rom_msa.coe** dans Notepad et fermez-le.
13. De retour dans la boîte de dialogue Select COE File, sélectionnez le fichier **rom_msa.coe** et cliquez sur **Open**.

Le champ Coefficients File du Distributed Memory Generator pointe maintenant sur le fichier d'initialisation **rom_msa.coe**.

14. Cliquez sur le bouton **Show** pour visualiser le contenu de ce fichier.
15. Vérifiez que le fichier contient 48 valeurs d'une taille de 4 bits chacune.
16. Dans l'espace COE Options du Distributed Memory Generator, sélectionnez **2** comme valeur du champ Radix.
17. Cliquez enfin sur **Generate** et vérifiez que le composant a bien été créé en inspectant les messages dans la Console.

Le symbole de la mémoire rom_msa est placé dans la librairie …\didact_sc\ipcore_dir.

## Placez le symbole de la ROM

Placez le symbole de la mémoire rom_msa dans le schéma msa_sc à l'aide de la fonction **Add Symbol** de la barre d'outils en effectuant les étapes suivantes :

1. Sélectionnez la librairie locale **U:\didact_sc\ipcore_dir** dans la liste Categories de l'onglet Design et sélectionnez le symbole rom_msa dans la liste Symbols.

2. De retour dans le schéma ouvert dans l'espace de travail, placez le symbole rom_msa dans le haut de la partie centrale du schéma (référez-vous à la Figure 29 pour disposer les composants).

### Ajoutez les autres composants au schéma

Vous devez ajouter des éléments de mémoire synchrones pour confectionner un registre d'état. Ce registre sert à maintenir la valeur de l'état en cours pendant toute la durée d'une période d'horloge. Ajoutez trois bascules D comme suit :

1. Sélectionnez **Add > Symbol** ou cliquez sur l'icône **Add symbol** de la barre d'outils.
2. Sélectionnez l'item **Flip_Flop** de la liste Categories.
3. Sélectionnez ensuite le type de bascule **fdc** dans la liste Symbols.
4. Repositionnez le pointeur de la souris dans l'espace de travail et placez trois instances de la bascule **fdc** à droite du composant rom_msa. Fiez-vous à la Figure 29 pour placer les bascules au bon endroit.

### Ajouter des bus et des taps

En mode schématique, un bus n'est rien d'autre qu'un fil qui se voit attribué un nom évoquant plusieurs bits. Par exemple, le fil monbus(3:0) est un bus de 4 bits. Vous pouvez ajouter des bus dans votre schéma de la même façon que vous avez tracé des fils, excepté que vous devez lui attribuer un nom évoquant plusieurs bits pour qu'il soit considéré comme un bus. Ajoutez deux bus de la façon suivante :

1. Sélectionnez **Add >Wire** dans le Navigateur de projet ou cliquer sur l'icône **Add Wire** dans la barre d'outil.
2. Cliquez sur l'entrée **a(5:0)** de la ROM, faites un angle droit et laisser prendre le bus un peu plus bas que le côté horizontal inférieur de la ROM, comme montré sur la Figure 25.
3. Nommez ce bus **romin(5:0)** à l'aide de la fonction **Add Net Name** de la barre d'outils.
4. Tracez aussi un bus pour la sortie **spo(3:0)** mais terminez-le un peu plus bas, comme montré sur la Figure 25.
5. Nommez ce bus **romout(3:0)**.

**Note :** Lorsque vous êtes en mode Add Wire, vous pouvez terminer une connexion n'importe où en double-cliquant à l'endroit souhaité.

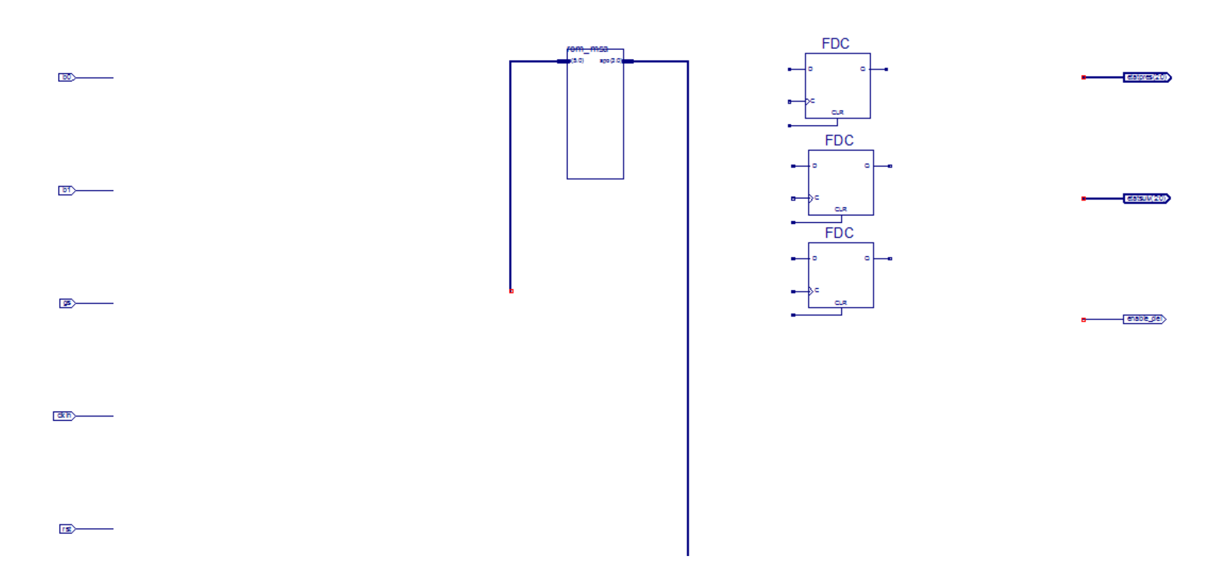



Figure 25. Tracez deux bus flottants de chaque côtés du symbole de la mémoire ROM.

6. Sélectionnez ensuite **Add > Bus Tap** dans le Navigateur de projet ou cliquer sur l'icône **Add Bus Tap** dans la barre d'outil.

Le curseur prend la forme d'un tap.

7. Positionnez le curseur de la souris sur le bus **romout(3:0)** et cliquez sur le bus.
8. Assurez-vous que les broches D des 3 bascules arrivent vis-à-vis le bus **romout(3:0).**
9. Dans l'espace Option, dans la partie gauche du Navigateur de projet, sous l'éditeur de symbole cliquez, sur **left**.

Remarquez que le champ Selected Bus Name contient romout(3:0) et que Net Name contient romout(3).

Pour tracer un tap à partir de la première bascule D :

10. Placer le bout du tap gauche apparaissant sur le curseur au bout de la broche D de la bascule et cliquez avec le bouton de gauche de la souris.

Un fil nommé romout(3) est tracé entre la broche D de la bascule et le bus.

Vous devriez obtenir le résultat montré à la figure suivante.

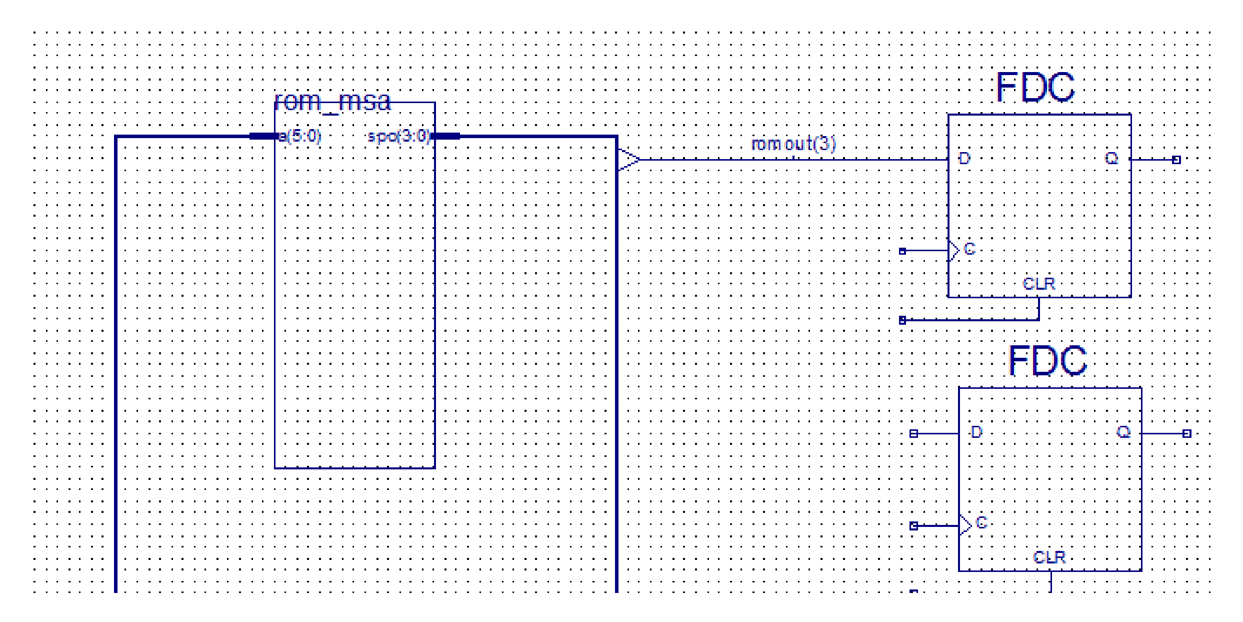



Figure 26. Un premier tap est ajouté entre la première bascule et le bus romout(3:0).

Remarquez que le Net Name dans l'espace Options est passé à romout(2).

11. Refaite la même chose pour tirer deux autres bus tap entre les broches D des deux dernières bascules et le bus. Les noms sont décrémentés automatiquement pour prendre la valeur romout(2) et romout(1), si ce n'est pas le cas utilisez la flèche droite ou gauche à côté du nom pour obtenir le bon fil.

## Complétez le schéma de la MSA

Complétez le schéma de la MSA en vous aidant du schéma complété montré à la Figure 29.

### Connectez les sorties de la MSA aux étiquettes de sortie

Suivez ces étapes :

1. Ajoutez des buffers vis-à-vis les broches Q des trois bascules D. Les buffers à ajouter correspondent à l'élément **buf** de la liste Symbols disponible dans la librairie **Buffer** de la liste Categories.

2. À l'aide de la fonction **Add Wire**, tracez un fil entre la sortie Q de chaque bascule et l'entrée du buffer qui lui fait face.

Ensuite, reliez la sortie des 3 buffers à l'étiquette etatpres(2:0). Pour ce faire procédez comme suit :

3. Déplacez l'étiquette etatpres(2:0) vers le bas pour qu'il arrive sous le buffer le plus bas.
4. À l'aide de la fonction Add Wire, faite courir un bus allant jusqu'au dessus de la sortie du buffer le plus haut. Le résultat devrait être conforme au schéma complété montré à la Figure 29.
5. Connectez la sortie de chaque buffer à ce bus à l'aide de la fonction **Add Bus Tap**, comme vous l'avez fait précédemment à la section Ajoutez des bus et des taps. Choisissez le sens Right. Nommez la sortie du 1er buffer etatpres(2); Nommez les deux autres etatpres(1) et etatpres(0) du plus haut au plus bas.
6. Ajoutez quatre buffers, sous les 3 bascules D, vis-à-vis la partie inférieure du bus romout(3:0). Consultez la Figure 29 pour savoir comment disposer les composants.
7. Connectez l'entrée de chaque buffer à un bit du bus romout(3:0) à l'aide de la fonction **Add Bus Tap**. Les bus tap doivent être positionnés vers la gauche (option left).
8. Assurez-vous que l'entrée du 1er buffer est connectée au fil romout(3) du bus, celle du 2ième au bit romout(2) et celle du 3ième, au but romout(1).
9. Ensuite, déplacez l'étiquette etatsuiv(2:0) légèrement sous le 3ième buffer (entre le 3ième et le 4ième buffer, comme montré sur le schéma complété, Figure 29).
10. Tracez un bus partant du bout de l'étiquette etatsuiv(2:0) vers le haut, jusqu'en haut du premier buffer.
11. Connectez la sortie de chaque buffer au bus etatsuiv(2:0) à l'aide de la fonction **Add Bus Tap**. Les tap doivent être positionnés vers la droite. Le bit etatsuiv(2) doit être connecté sur la sortie du 1er buffer, etatsuiv(1) sur celle du 2ième buffer et etatsuiv(0) sur celle du 3ième buffer.
12. Ensuite, déplacez l'étiquette enable_del vis-à-vis le dernier buffer, sous l'étiquette etatsuiv(2:0).
13. Connectez l'entrée de ce buffer au bus romout(3:0) grâce à la fonction **Add Bus Tap** avec le tap positionné vers la gauche. Après avoir cliqué sur le bus avec le tap du curseur, changez le Net Name pour romout(0) dans l'espace Options. Cliquez sur l'entrée du buffer avec le curseur pour créer le tap. Le bit créé doit prendre le nom romout(0).
14. À l'aide de la fonction **Add Wire**, complétez ce chemin en traçant un fil entre la sortie du dernier buffer et l'étiquette enable_del.

### Connectez les entrées de la MSA aux étiquettes d'entrées

Maintenant que toutes les sorties de la MSA sont connectées aux étiquettes du schéma, vous allez connecter les étiquettes d'entrées au entrées de la MSA. Procédez comme suit :

1. Tout d'abord, ajoutez trois buffers vis-à-vis la partie inférieure du bus romin(5:0) comme illustré sur la Figure 29.
2. À l'aide de la fonction **Add Wire**, rapportez les bits d'états présents (les sorties des bascules D) vers l'entrée de la ROM, dans le but de les connecter ensemble; Amenez chaque bit d'état vis-à-vis du bus romin(5:0) comme montré sur la figure suivante.

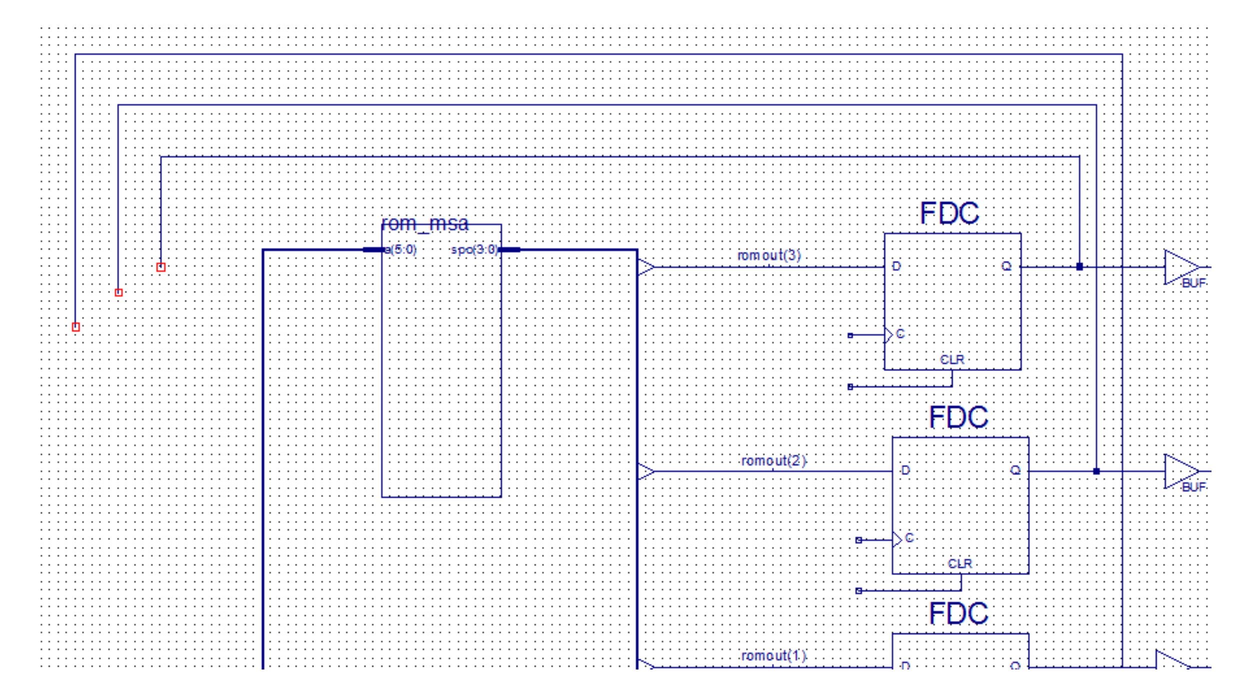



**Figure 27. Rapportez le bus des bit des états présents (bus etatpres(2:0)) à l'entrée de la ROM vis-à-vis du bus romin(5:0) dans le but de les connecter ensuite.**

3. Ensuite, connectez chaque fil au bus romin(5:0) grâce à la fonction **Add Bus Tap**; Positionnez le tap vers la droite; Commencez par le fil le plus haut; Après, vérifiez que le 1$^{er}$ fil se nomme romin(5), le 2$^{ième}$ romin(4) et le 3$^{ième}$ romin(3).
4. Sous les trois connexions créées, placez trois buffers pour connecter les étiquettes d'entrées b0, b1 et gs.

5. Ensuite, placez les étiquettes b1, b0 et gs, de bas en haut, vis-à-vis les trois buffers comme montré sur la figure suivante; Déplacez les étiquettes à l'aide d'un clique glissé de souris pour les placer comme montré sur la figure suivante; Attention de ne pas intervertir les étiquettes b0 et b1, sinon la MSA ne produira pas le comportement souhaité.

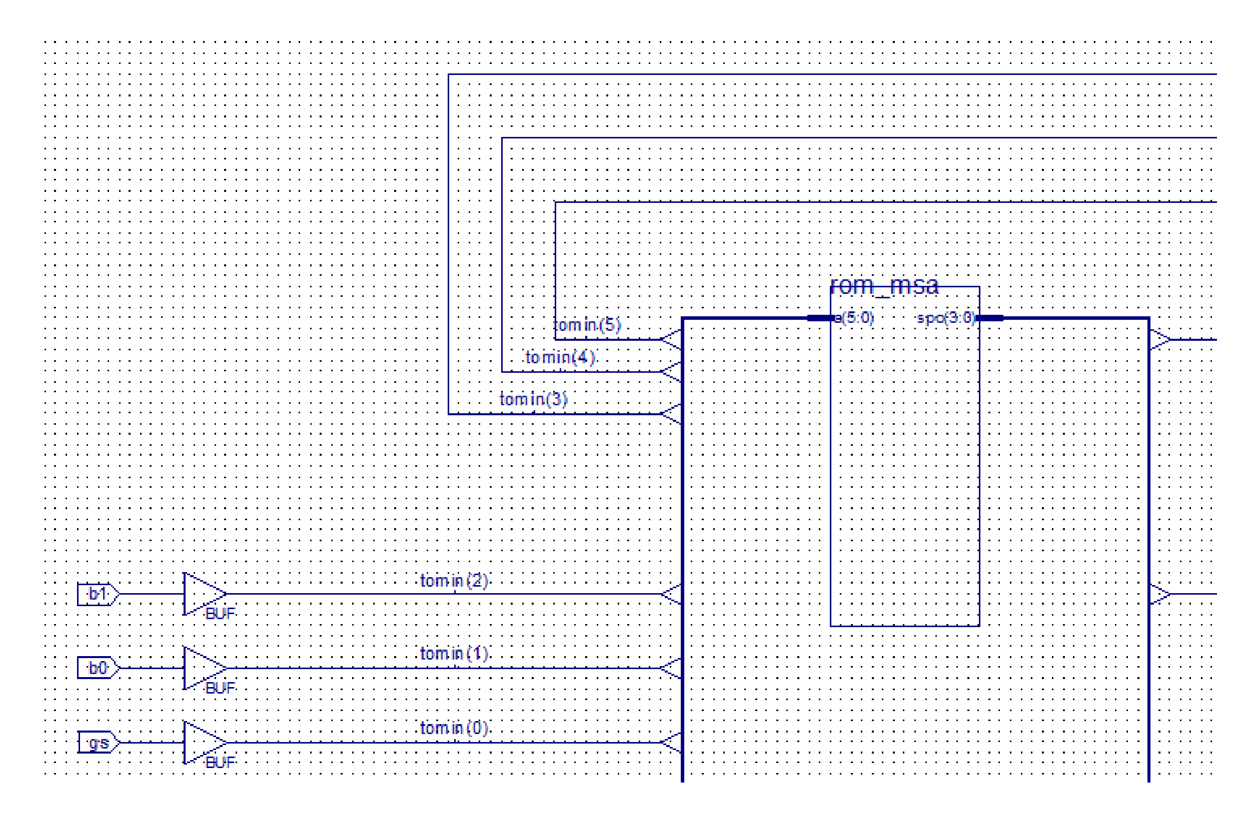



Figure 28. Placez les 3 étiquettes b0, b1 et gs vis-à-vis le bus romin(5:0) et insérez trois buffer entre ceux-ci.

6. Connectez les trois étiquettes aux entrées des buffers qui leur font face à l'aide de la fonction **Add Wire**.
7. Connectez la sortie des trois buffers au bus romin(5:0) à l'aide de la fonction **Add Bus Tap** : Positionnez le tap vers la droite et inscrivez romin(2) dans le champ **Net Name**; Cliquez sur la sortie du 1$^{er}$ buffer pour créer le tap; Assurez-vous que le fil créé se nomme romin(2).
8. Avant de créer les deux derniers tap, assurez-vous que les fils qui seront créés se nommeront romin(1) et romin(0) : inscrivez ces noms au préalable dans le champ **Net Name** avant de créer chaque tap.
9. Assurez-vous que l'étiquette b1 se reporte sur romin(2), b0 sur romin(1) et gs sur romin(0).

## Renommez les instances du schéma msa_sc

1. Renommez toutes les instances du schéma msa_sc comme suit :
   - La mémoire ROM : **rom_inst**.
   - Les trois bascules D (de haut en bas): **Q2_inst, Q1_inst** et **Q0_inst**.
2. Assurez-vous que votre schéma est conforme au schéma complété montré à la Figure 29 et sauvegardez-le.
3. Effectuez ensuite un DRC en cliquant sur **Tools > Check Schematic** et assurez-vous de corriger les erreurs et les warnings annotés dans la console suite au DRC.
4. Le schéma de la MSA est terminé. Fermez le fichier msa_sc.

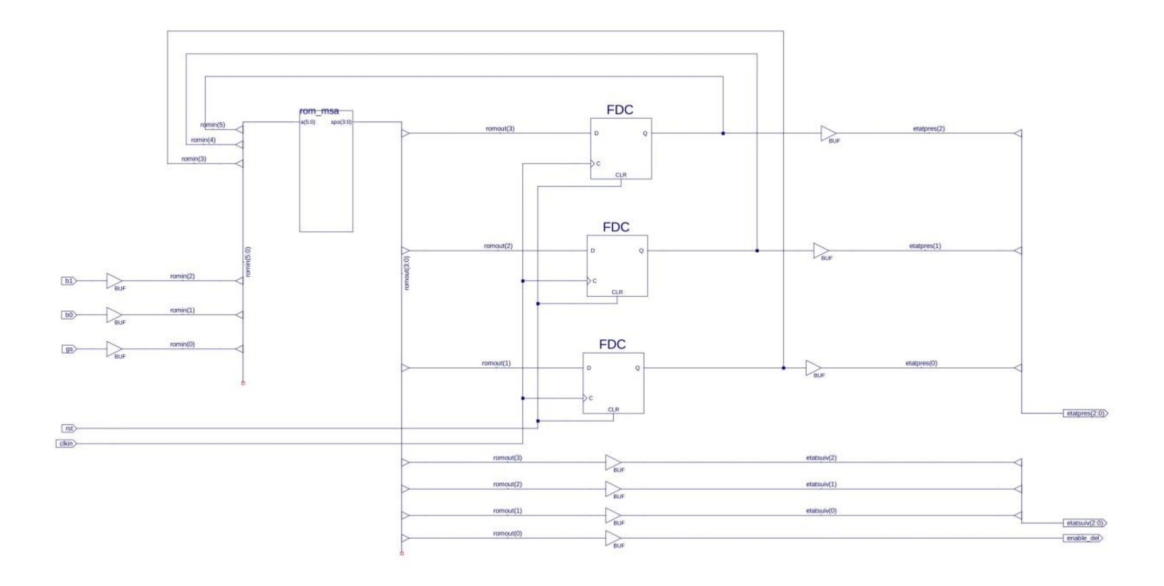



Figure 29. Schéma complété de la MSA.

## Créez le symbole du schéma de la MSA

Créez un symbole pour la MSA et placez-le dans le schéma top level comme vous l'avez fait pour les autres modules:

4. Tout d'abord, sélectionnez l'onglet **Design** à gauche du Navigateur de projet et sélectionnez l'item **msa_sc** de l'arborescence du projet, dans l'espace Hierarchy.

5. Déployer la liste d'item Design Utilities en cliquant sur le symbole **+** lui faisant face.
6. Double-cliquez sur l'item **Create Schematic Symbol**.

### Placez le symbole msa_sc

Placer le symbole de la MSA dans le schéma top level en procédant comme suit :

7. Double-cliquez sur l'item **didact_top** dans l'arborescence du projet.
8. Sélectionnez **Add > Symbol** ou cliquez sur l'icône **Add symbol** de la barre d'outils.
9. Dans la liste Categories, sélectionnez la librairie de symboles locale **…/didact_sc**.
10. Sélectionnez le symbole **msa_sc** dans la liste Symbols et placez-le dans le schéma top level, à droite du symbole de l'encodeur, comme montré sur le schéma complété, reproduit à la Figure 7.
11. Renommez l'instance de ce nouveau module en tant que **msa_inst**.
12. Sauvegardez le schéma top level.

## Création d'un registre à décalage

Ajoutez un registre à décalage qui permettra d'activer une série de DEL en donnant une impression de défilade. La sortie enable_del activera le registre quand la bonne séquence sera détectée. Procédez comme suit :

1. Sélectionnez **Add > Symbol** ou cliquez sur l'icône **Add symbol** de la barre d'outils.

L'éditeur de symbole s'ouvre à gauche du Navigateur de projet.

2. Sélectionnez l'item **Shift_Register** de la liste Categories.
3. Sélectionnez ensuite le registre **SR8CE** dans la liste Symbols.
4. Placez le symbole du registre sous le symbole de la MSA, comme montré sur le schéma complété, Figure 7.
5. Double-cliquez sur le symbole du registre à décalage que vous venez de placer.
6. Inscrivez **shiftreg_inst** dans le champ InstName.

Tous les composants du schéma top level sont maintenant placés.

## Ajoutez les étiquettes d'entrées/sorties

Ajouter cinq étiquettes d'entrées au schéma top level : rst, clkin, bouton1, bouton2 et bouton3. Pour ajouter ces étiquettes :

1. Dessinez des fils flottant partant des broches RST_IN et CLKIN_IN du module dcm1 et des broches sig_in appartenant aux trois modules anti-rebond.
2. Ensuite, sélectionnez **Add > Net Name**.
3. Tapez **rst** dans le champ Name de l'espace Options.
4. Cliquez sur la partie la plus à gauche du fil flottant connecté à la broche RST_IN du module dcm1.

Le fil flottant est renommé rst.

5. Répéter les étapes 2, 3 et 4 pour les noms de fils clkin, bouton1, bouton2 et bouton3, comme illustré sur la Figure 30.

Après que tout les fils flottants aient été identifiés, créez les étiquettes :

6. Sélectionnez **Add > I/O Markers**.
7. Cliquez dans en haut à gauche du fil nommé rst et glisser la souris de haut en bas en maintenant le bouton de la souris enfoncé pour sélectionné les cinq fils ainsi que leur noms, comme illustré sur la Figure 30.

Ensuite, ajoutez une étiquette de sortie Q_del(7:0) connectée à la sortie du registre à décalage comme suit :

1. Dessinez un bus flottant à la sortie à la sortie du registre à décalage à l'aide de la fonction **Add Wire**.
2. Nommez ce bus Q_del(7:0) à l'aide de la fonction **Add Net Name**.
3. Créez une étiquette Q_del(7:0) comme vous l'avez fait précédemment à l'aide de la fonction **Add I/O Markers** et de la souris en sélectionnant le bus Q_del(7:0).

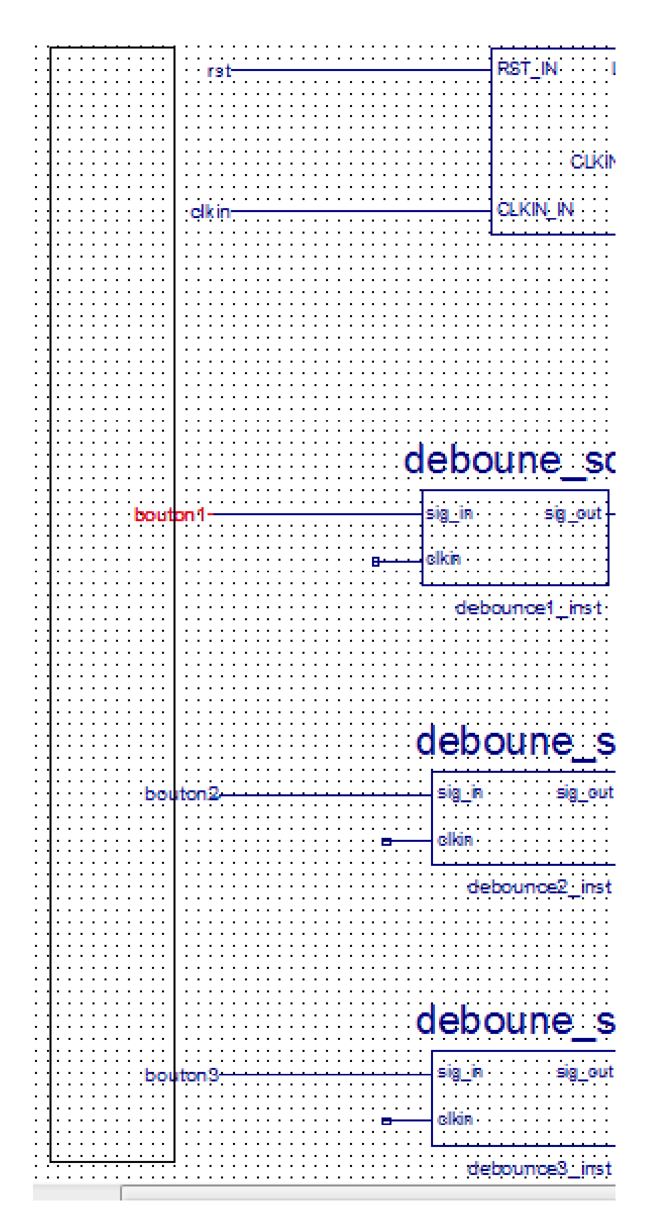



Figure 30. Création des étiquettes d'entrées à l'aide de la fonction Add I/O Markers et de la souris.

## Complétez le schéma top level

Complétez le schéma didact_top en réalisant toutes les connexions entre les symboles que vous avez placés. Effectuez les étapes suivantes pour fignoler le schéma et référez-vous au schéma complété (Figure 7) pour vous aider dans cette tâche. Chaque item de la liste de tâches à effectuer

a déjà été couvert dans les sections précédentes de ce didacticiel. Référez-vous à ces sections pour de plus amples détails sur les façons de les accomplir si nécessaire.

Effectuez les actions suivantes pour compléter le schéma top level :

1. Dessinez un fil entre la broche clk_out1 du module dcm1 et la broche clkin du module diviseur_clk et nommez-le **clk_dcm1**.
2. Dessinez trois fils flottant à partir des broches clk2hz, clk16hz et clk2khz du module diviseur_clk et nommez-les **clk2hz, clk16hz et clk2khz** respectivement.
3. Dessinez des fils flottants à partir des broches clkin des trois modules debounce_sc et nommez-les **clk2khz**.

**Notez que l'éditeur de schéma considère que deux fils ayant le même nom sont physiquement connectés**.

4. Tracez un fil entre la broche sig_out de debounce1_inst et la broche E1 de decodeur_sc, un autre entre la broche sig_out de debounce2_inst et la broche E2 de decodeur_sc et un dernier entre la broche sig_out de debounce3_inst et la broche E3 de decodeur_sc.
5. Nommez respectivement ces fils **debout1, debout2 et debout3**.
6. Tracez un fil entre la broche O0 de l'encodeur et la broche b0 de msa_sc, un autre entre la broche O1 de l'encodeur et la broche b1 de msa_sc et un dernier entre la broche GS de l'encodeur et la broche gs de msa_sc.
7. Nommez respectivement ces fils **b0, b1 et gs**.
8. Tracez des fils flottants partant des broches clkin et rst de msa_sc. Nommez-les **clk_dcm1** et **rst**.
9. Tracez un bus flottant à partir de la broche etatsuiv(2:0) de msa_sc et nommez-le **etatsuiv(2:0)**.
10. Tracez un fil flottant à partir le la broche enable_del de msa_sc et nommez-le **enable_del**.
11. Tracez un bus flottant à partir de la broche etatpres(2:0) de msa_sc et nommez-le **etatpres(2:0)**.
12. Tracez un fil flottant partant de la broche d'entrée CE du registre à décalage et nommez-le **enable_del**.
13. Tracez des fils flottants à partir des broches SLI, C et CLR du registre à décalage. Nommez respectivement ces fils **clk2hz**, **clk16hz** et **rst**.

Le schéma top level est maintenant complété.

Sauvegardez le schéma et effectuez un DRC à l'aide de la fonction Check Design Rules. Vous devriez obtenir les deux warnings suivants :

```
WARNING:DesignEntry:221 - didact_top.sch Bus 'etatsuiv(2:0)' is connected to source pins and/or I/O markers, but not connected to any load pins or I/O marker.

WARNING:DesignEntry:221 - didact_top.sch Bus 'etatpres(2:0)' is connected to source pins and/or I/O markers, but not connected to any load pins or I/O marker.
```

Laissez ces warnings tels quels. Corrigez les erreurs et les autres warnings s'il y lieu.

Fermez le schéma top level. Vous avez complété la conception du détecteur de séquence avec l'approche par schéma.

Pour poursuivre avec  les étapes subséquentes de l'approche par schéma :

- Allez au chapitre Simulation fonctionnelle avec Isim pour effectuer une simulation pré-synthèse du système.
- Ou allez au chapitre Implémentation du projet pour effectuer le placement routage du projet.

## Approche de conception par langage HDL

Cette partie du didacticiel porte sur la conception du détecteur de séquence à l'aide de l'approche de conception FPGA par langage HDL. Dans cette partie, le projet du détecteur de séquence présente une structure hiérarchique reposant sur des modules décris en langage VHDL. En effet, le fichier top level du projet permet de représenter l'ensemble du système à l'aide d'une description VHDL structurelle faisant appel à des composants de niveaux hiérarchiques inférieurs. Ces composants sont divers natures dont des modules VHDL, des IP générés à l'aide de l'Architecture Wizard ou des modules paramétrés générés à l'aide du CORE Generator.

Cette section vous permettra de compléter la conception du détecteur de séquence à l'aide d'une description VHDL. Ensuite, vous serez en mesure de poursuivre avec les étapes subséquentes du didacticiel, soit la simulation fonctionnelle, l'implémentation du projet, la simulation avec timings et la configuration du FPGA avec le système conçu.

### Création d'un nouveau projet dans ISE

Pour débuter cette partie du didacticiel, créez un nouveau projet ISE ayant pour cible le FPGA Spartan 3E.

1. Pour lancer la suite ISE version 12.4, double-cliquez sur l'icône du **Navigateur de projet ISE**, ou sélectionnez **Start > Programs > Xilinx ISE design suite 12.4 > ISE Design Tools > xx-bit Project Navigator** alternativement.




Figure 31. Icône du Navigateur de projet ISE 12.4 disponible sur le bureau.

2. Dans le Navigateur de projet ISE, cliquez sur **File > New Project**, ou cliquez sur le bouton **New Project** dans la fenêtre de gauche.
3. Tapez **didact_hdl** comme nom de projet.

Notez que didact_hdl est ajouté au chemin initial contenu dans le champ Location.

4. Sélectionnez **HDL** pour le champ Top-Level Source Type et cliquez sur **Next**.
5. Entrez les valeurs suivantes dans la fenêtre New Project Wizard – Device properties :
   - Product Category: All
   - Family: Spartan6
   - Device: XC6SLX16
   - Package: CSG324
   - Speed: -3
   - Synthesis Tool: XST (VHDL/Verilog)
   - Simulator: ISim VHDL
   - Preferred Language: VHDL

La Figure 32 montre la bonne configuration à avoir pour cette boîte de dialogue.

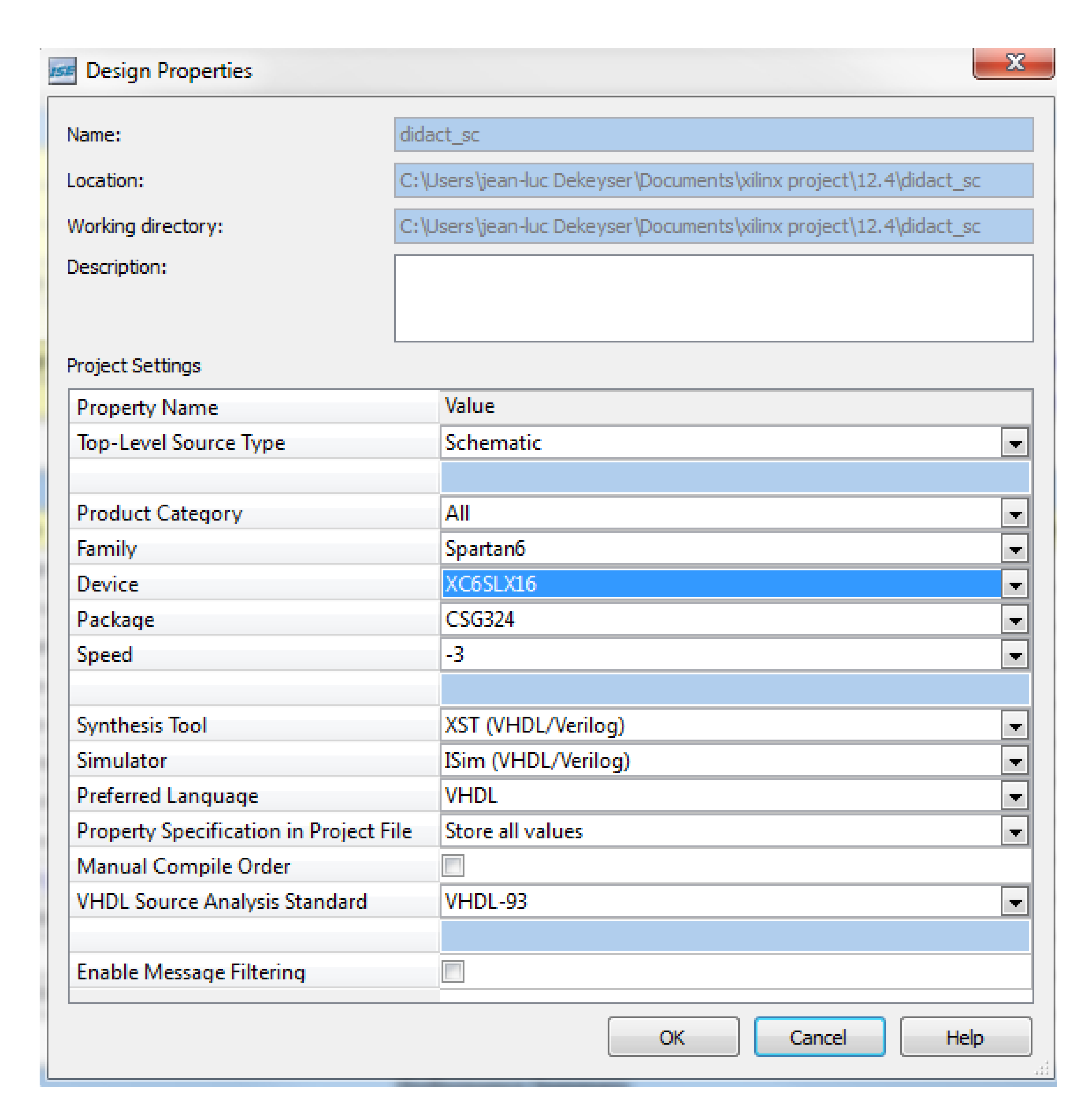



**Figure 32. Fenêtre New Projet Wizard – Device properties: paramètres à entrer pour la configuration du FPGA Spartan 6 utilisé pour ce didacticiel.**

6. Cliquez successivement sur **Next** et **Finish** dans les fenêtres Project Summary du New Project Wizard se succédant.

À ce stade, vous obtenez un nouveau projet ISE ne contenant encore aucun fichier HDL ou module IP.

## Création d'un fichier VHDL top level

Créez un nouveau fichier top level de la façon suivante :

1. Assurez-vous que l'onglet Design est sélectionné dans l'espace Hierarchy situé dans la partie supérieure gauche du Navigateur de Projet et cliquer sur **Project > New Source** dans la barre de menu.
2. Sélectionnez **VHDL Module** dans la fenêtre *New Source Wizard – Select Source Type*.
3. Tapez **didact_top** comme nom de fichier et vérifiez que la case **Add to project** est cochée.
4. Cliquez sur **Next.**
5. Dans la boîte de dialogue Define Module, entrez cinq ports d'entrées et un port de sortie sous Port Name, comme montré à la Figure 33 :
    a. Entrez **rst, clkin, bouton1, bouton2** et **bouton3** dans les cinq première cases.
    b. Entrez **Q_del** dans 6$^{ième}$ case et changez le champ Direction à **out**.
    c. Toujours pour le port Q_del, cochez le champ **Bus**, entrez **7** sous le MSB et **0** sous le champ LSB.
6. Cliquez sur **Next**, et enfin sur **Finish**.

Remarquez que l'élément didact_top est ajouté au projet dans l'espace Hierarchy de l'onglet Design.

De plus, le fichier vide didact_top.vhd s'ouvre dans l'espace de travail comme montré à la Figure 34. Ce fichier VHDL est divisé en deux parties importantes : l'entité du module (située entre les balises `entity didact_top is` et `end didact_top;` ) et l'architecture du module (située entre les balises `architecture Behavioral of didact_top is` et `end Behavioral;`). L'entité contient une liste des ports d'entrées/sorties du module VHDL accessibles au niveau hiérarchique supérieur, alors que l'architecture décrit le fonctionnement du module. L'entité du module top level est complète alors que son architecture est encore vide. Vous devrez la compléter par la suite.

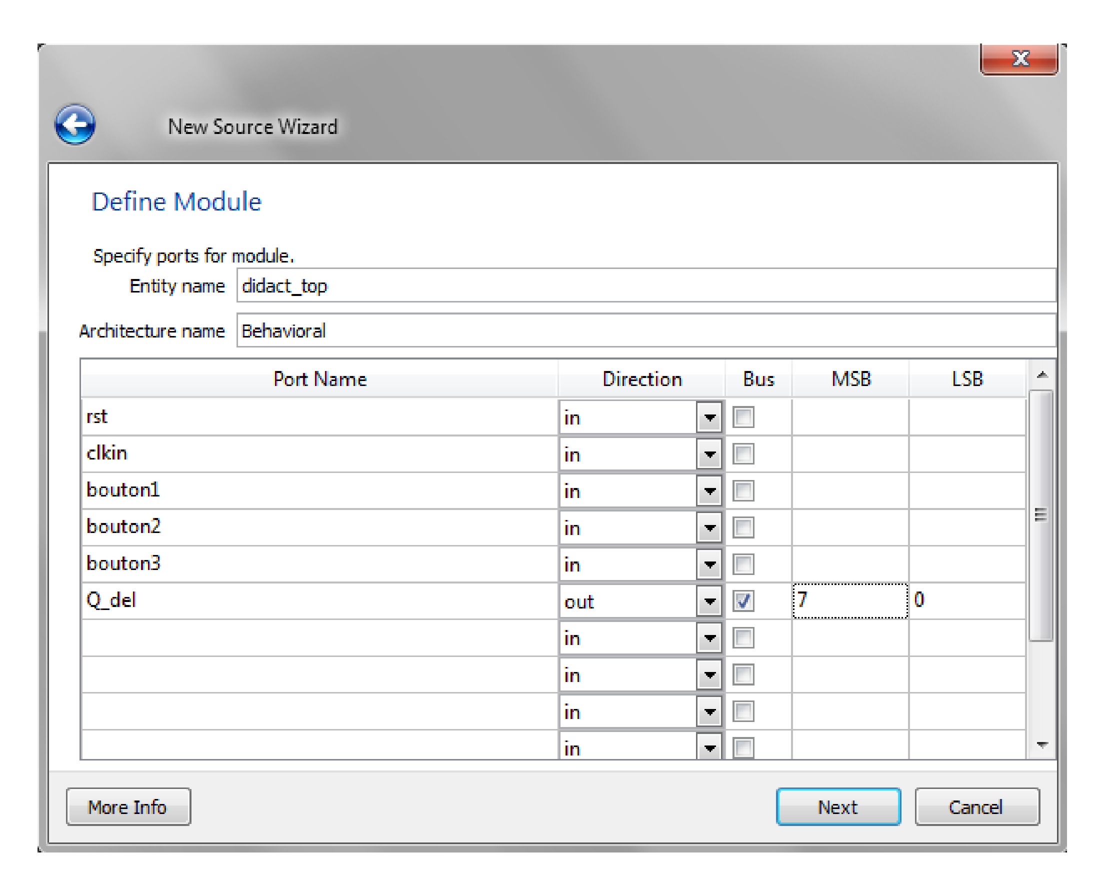



Figure 33. La boîte de dialogue Define Module pour entrer les ports d’entrées/sorties du module didact_top.

Pour la suite de cette partie, la démarche adoptée consiste à créer les sous modules VHDL et IP nécessaires au projet et à les instancier et les interconnecter ensembles dans le fichier top level didact_top.vhd. Référez-vous au Tableau 3 pour en savoir plus sur chaque sous module.

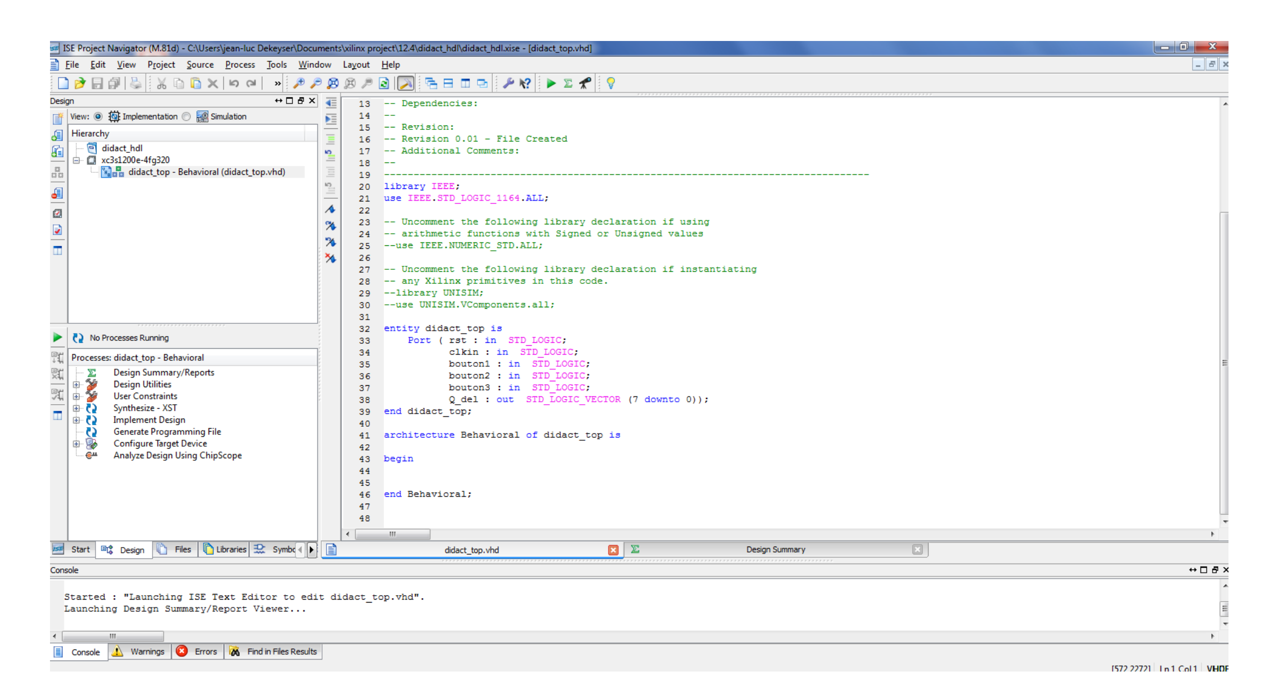



**Figure 34. Le fichier VHDL top level didact_top.vhd doit être complété.**

## Création d'un module VHDL anti-rebond

Les boutons poussoirs et les commutateurs disponibles sur la carte Nexys2 ne disposent pas de circuits d'anti-rebond. Par conséquent, les rebonds doivent être traités à l'aide du FPGA. Pour cette partie du didacticiel, vous allez créer un module VHDL d'anti-rebond pour traiter les trois boutons utilisés pour entrer la séquence.

Les étapes suivantes montrent comment créer un module VHDL d'anti-rebond à l'aide du New Source Wizard du Navigateur de projet ISE. Comme pour le module top level, le module d'anti-rebond sera décrit dans un fichier VHDL composé d'une entité décrivant ses ports d'entrées/sorties et d'une architecture décrivant son fonctionnement.

Vous créerez d'abord un fichier VHDL vide et vous utiliserez les gabarits VHDL disponibles dans le Navigateur ISE pour compléter la description de ce module.

Pour créer le module VHDL d'anti-rebond:

1. Sélectionnez **Project > New Source.**

2. Sélectionnez **VHDL Module** dans la boîte de dialogue Select Source Type et entrez **debounce_hdl** comme nom de fichier.
3. Vérifiez que la case **Add to project** est cochée et cliquez sur **Next.**
4. Dans la boîte de dialogue Define Module, entrez deux ports d'entrées et un port de sortie sous Port Name, comme montré à la Figure 35 :
   a. Dans les 3 premières cases, entrez **sig_in, clkin** et **sig_out**.
   b. Changez le champ Direction à **in** pour sig_in et clkin et à **out** pour sig_out.
   c. Ne cochez pas les champs Bus.
5. Cliquez sur **Next**, et enfin sur **Finish**.

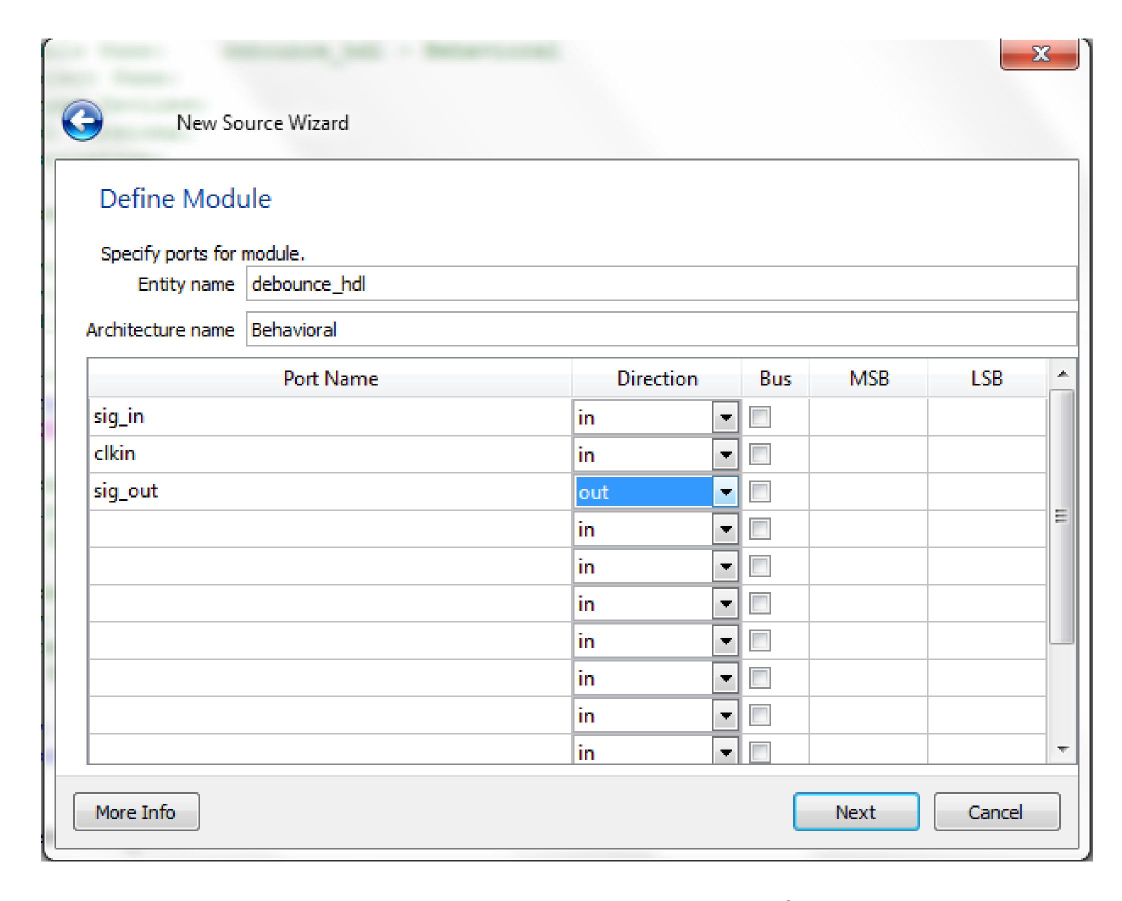



Figure 35. La boîte de dialogue Define Module pour entrer les ports d'entrées/sorties du module debounce_hdl.

Le fichier VHDL vide décrivant le module debounce_hdl s’ouvre dans l’espace de travail comme montré à la Figure 36. Ce fichier contient l’entité du module VHDL et son architecture vide à compléter.

Assurez-vous que l’onglet Design de l’espace Hierarchy est sélectionné et que le champ View pointe vers **Implementation**. Remarquez que l’item debounce_hdl est ajouté au projet dans l’espace Hierarchy. Il n’est pas encore un sous module de didact_top.

Vous allez compléter le fichier debounce_hdl.vhd grâce aux étapes suivantes.

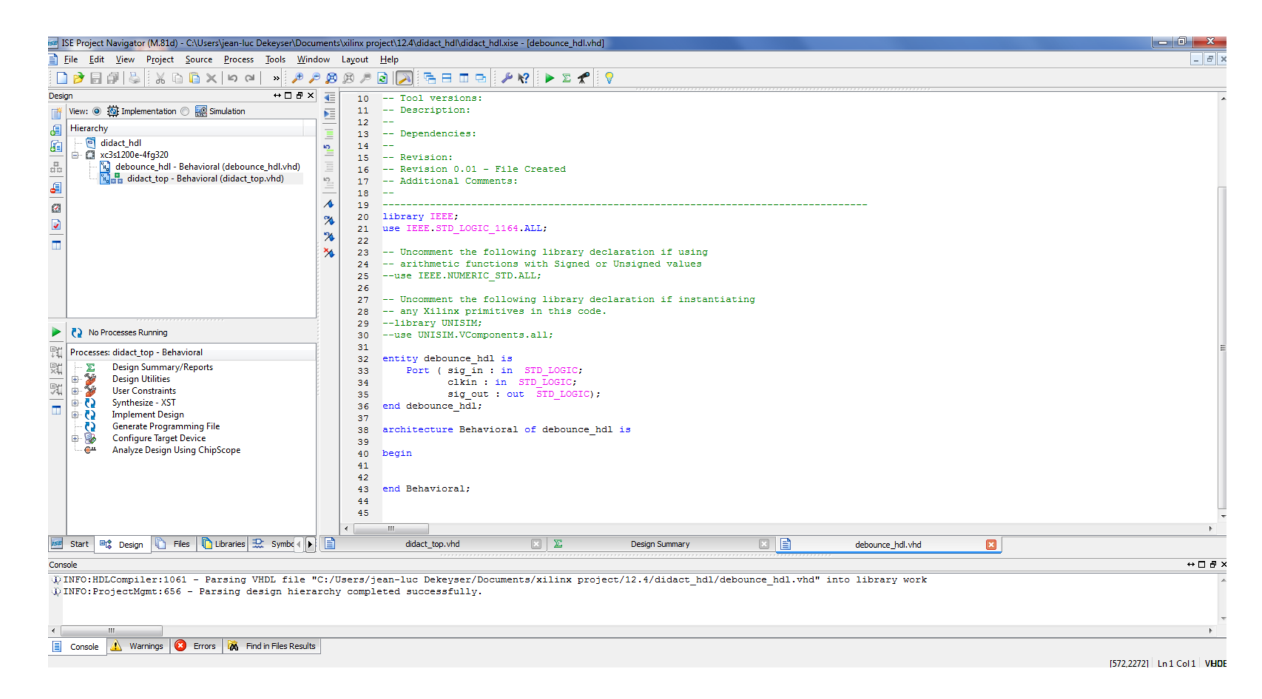



**Figure 36. Fichier VHDL vide debounce_hdl.vhd.**

## Utilisation des gabarits VHDL

ISE comprend plusieurs gabarits VHDL représentants une foule de composants logiques couramment utilisés comme des compteurs, des multiplexeurs ou des bascule D. Vous allez utiliser le gabarit ISE de circuit d’anti rebond (Debounce circuit) pour compléter l’architecture du module debounce_hdl. Effectuez les étapes suivantes :

1. Dans le Navigateur de projet, sélectionnez **Edit > Language Templates**.
2. Déployez l’item **VHDL** de la liste en cliquant sur le **+**.
3. Sous VHDL, déployez l’item **Synthesis Constructs**, déployez l’item **Coding examples**, déployez l’item **Misc** et sélectionnez le gabarit VHDL **Debounce circuit**, comme illustré à la Figure 37.

Le code VHDL du gabarit du circuit d’anti-rebond s’ouvre dans l’espace de travail à droite.

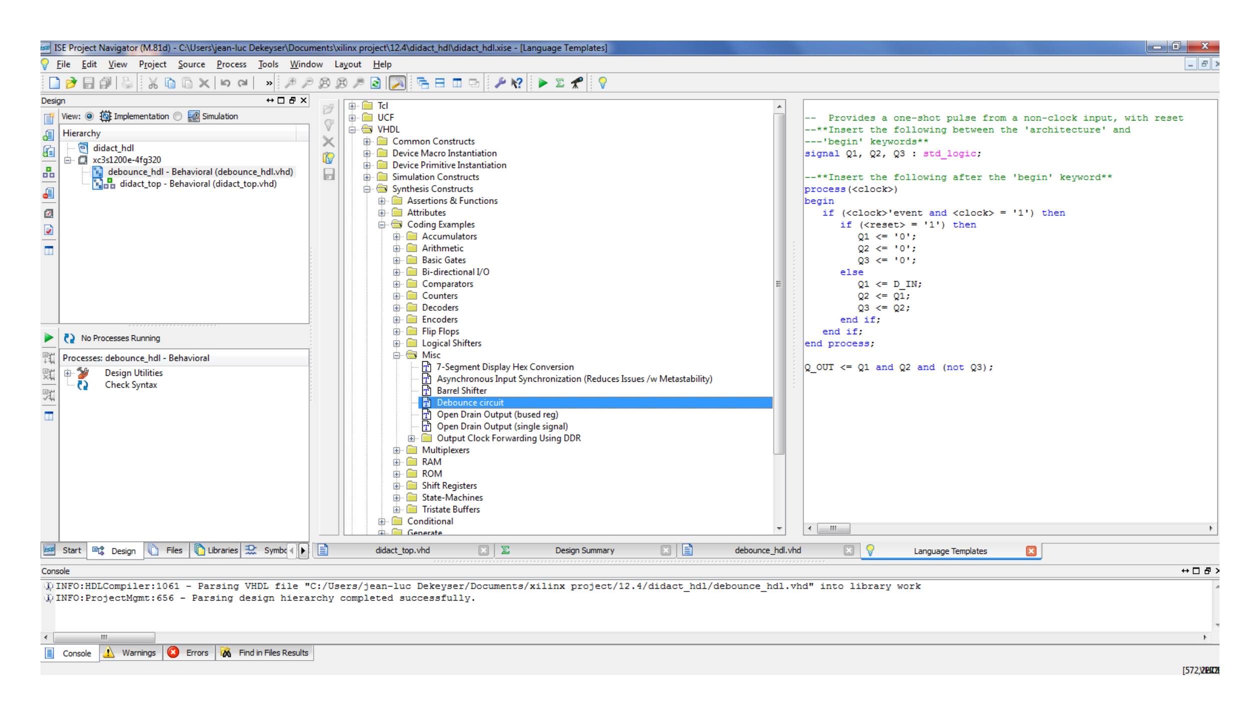



**Figure 37. Le gabarit VHDL Debounce circuit s’affiche dans l’espace de travail à droite. Utilisez ce gabarit pour compléter le fichier VHDL debounce_hdl.vhd.**

## Ajoutez le gabarit VHDL à votre fichier

Ajoutez le gabarit à votre fichier à l’aide de la méthode « Use in file » de la façon suivante :

1. Ouvrer ou afficher le fichier debounce_hdl dans l’espace de travail, si ce n’est pas déjà fait.
2. Dans ce fichier, effacez la ligne de code `begin`, juste sous la ligne `architecture Behavioral of debounce_hdl is` et laissez le curseur juste sous cette ligne.
3. Revenez à l’onglet Language Templates dans l’espace de travail.
4. Cliquez avec le bouton droit de la souris sur **Debounce circuit** et sélectionnez **Use in file** dans le menu déroulant, comme illustré à la Figure 38.

5. Fermez le Language Templates.
6. Ouvrez le fichier debounce_hdl.vhd pour vérifier que le code du gabarit VHDL y a bien été ajouté.
7. Complétez le fichier VHDL du module debounce_hdl en effectuant les étapes suivantes :
    a. Remplacez les **<clock>** par **clkin**, le **D_IN** par **sig_in** et le **Q_OUT** par **sig_out**.
    b. Placez le curseur après la ligne `signal Q1, Q2, Q3 : std_logic;` et appuyez sur Enter ↵ deux fois.
    c. Inscrivez la ligne de code `begin` à cet endroit.
    d. Enlevez le reset du gabarit VHDL en supprimant les cinq lignes de code situées sous la ligne `if (clkin'event and clkin = '1') then`, soit celles qui commencent par `if (<reset> = '1') then` et qui se terminent par `else`, et supprimez un des `end if;`
    e. Réajustez l'indentation du fichier au besoin.

    Le fichier VHDL debounce_hdl.vhd est complété. Vérifiez que votre fichiez est conforme au fichier original debounce_hdl.vhd placé à l'Annexe 4 de ce document.

8. Sauvegardez le fichier en sélectionnant **File > Save** et fermez-le.
9. Sélectionnez l'item **debounce_hdl** dans l'onglet **Design** à gauche du Navigateur de projet de l'arborescence du projet (dans l'espace Hierarchy).
10. Double-cliquez sur l'item **Check syntax** dans l'espace Processes (sous Hierarchy).
11. Corriger les erreurs de syntaxe affichées dans la Console s'il y a lieu et fermer le fichier debounce_hdl.vhd.

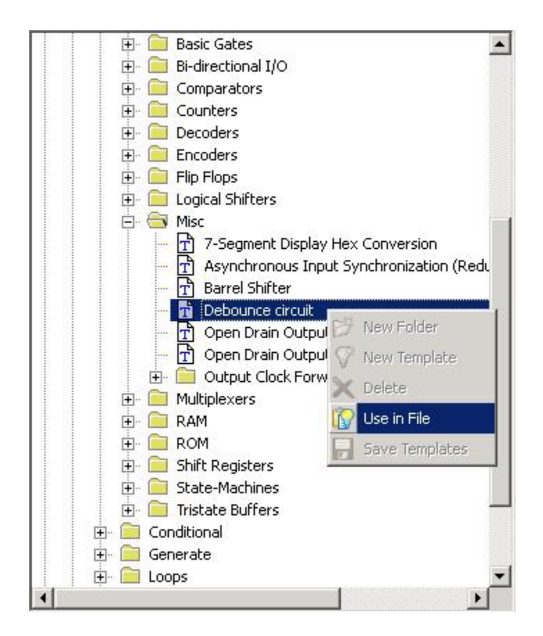



**Figure 38. La fonction Use in File du Language Templates permet d'insérer un gabarit VHDL dans un fichier vide créé au préalable.**

## Déclarez et instanciez le module d'anti-rebond

Vous allez déclarer le composant debounce_hdl dans le fichier top_level et instancier trois de ces sous modules. Chacun de ces sous modules sera chargé de traiter un des signaux (bouton1…3) activés par les boutons poussoirs. Suivez les étapes suivantes :

1. Dans l'espace Hierarchy du Navigateur de projet, double-cliquez sur l'item **didact_top**. Le fichier du schéma top level s'ouvre dans l'espace de travail.
2. Ouvrez maintenant le fichier debounce_hdl, en déroulant Design Utilities de la fenêtre Proccesses, vous allez créer une instantiation du composant automatiquement debounce_hdl.vhi. Il suffit de cliquer view HDL Instantiation Template.
3. Par un copier coller, déclarez ensuite le composant debounce_hdl comme suit :
   - Sous la ligne `architecture Behavioral of didact_top is (en haut du begin)`, ajoutez ces lignes de code :

```
component debounce_hdl
port(
     sig_in : in  STD_LOGIC;
     clkin : in  STD_LOGIC;
     sig_out : out  STD_LOGIC);
```

```
end component;
```

4. Déclarez tous les signaux internes d'interconnexions du module didact_top dès maintenant comme suit :
    - Sous les lignes de code que vous venez d'ajouter (en haut du `begin`), ajoutez ces lignes de code :

```
signal clk_dcm1 : std_logic;
signal CLKIN_IBUFG_OUT : std_logic;
signal CLK0_OUT : std_logic;
signal LOCKED_OUT : std_logic;
signal b0, b1, gs, enable_del : std_logic;
signal clk2hz,clk16hz,clk2khz : std_logic;
signal debout1,debout2,debout3 : std_logic;
signal shreg : std_logic_vector(7 downto 0);
```

**Note:** Tous les signaux d'interconnexions utilisés dans didact_top sont déclarés ici pour des raisons pratiques, bien que certains n'aient pas de liens directs avec le module debounce_hdl. Ces signaux serviront à interconnecter tous les modules que vous créerez dans cette partie.

5. Instanciez trois modules debounce_hdl en utilisant la deuxième partie du template de debounce_hdl.vhi avec un copier et trois coller puis renommez les instances et interconnectez-les avec les signaux internes et ports d'entrées/sorties. Procédez comme suit :
    - Sous le `begin`, ajoutez ces trois `port map` :

```
inst1_debounce: debounce_hdl port map(
      sig_in => bouton1,
      sig_out => debout1,
      clkin => clk2khz
);

inst2_debounce: debounce_hdl port map(
      sig_in => bouton2,
      sig_out => debout2,
      clkin => clk2khz
);

inst3_debounce: debounce_hdl port map(
      sig_in => bouton3,
      sig_out => debout3,
      clkin => clk2khz
);
```

6. Sauvegardez le schéma top level en sélectionnant **Save All**.
7. Fermez le fichier debounce_hdl.vhi.

Remarquez que les trois instances debounce_hdl ajoutées au fichier top level apparaissent maintenant sous l'item didact_top dans l'arborescence du projet.

8. Sélectionnez l'item **didact_top** dans l'onglet **Design** à gauche du Navigateur de projet de l'arborescence du projet (dans l'espace Hierarchy).
9. Dans l'espace Processes (sous Hierarchy), déployez l'item **Synthesize – XST**.
10. Double-cliquez sur l'item **Check syntax** de cette liste.
11. Corriger les erreurs de syntaxe affichées dans la Console s'il y a lieu et fermer le fichier didact_top.vhd.

## Création d'un module de gestion d'horloge

Le Architecture Wizard de ISE permet de créer et de configurer graphiquement une panoplie de module IP très rapidement. Vous allez créer et configurer un module de gestion d'horloge (DCM) qui dérivera une horloge d'une fréquence de 16 MHz à partir de l'horloge de référence de 100 MHz disponible sur la carte Spartan 6.

7. Cliquer sur **Project > New Source** dans la barre de menu du Navigateur de Projet ISE.
8. Sélectionnez **IP (CORE Generator & Architecture Wizard)** dans la fenêtre *Project Wizard – Select Source Type*.
9. Tapez **dcm1** comme nom de fichier, vérifiez que la case **Add to project** est cochée et cliquez sur **Next**.
10. Sélectionnez **FPGA Features and Design > Clocking > Clocking Wizard** dans l'onglet View by fonction de la boîte de dialogue Select IP, comme illustré à la Figure 17.
11. Cliquez sur **Next** et ensuite sur **Finish**.

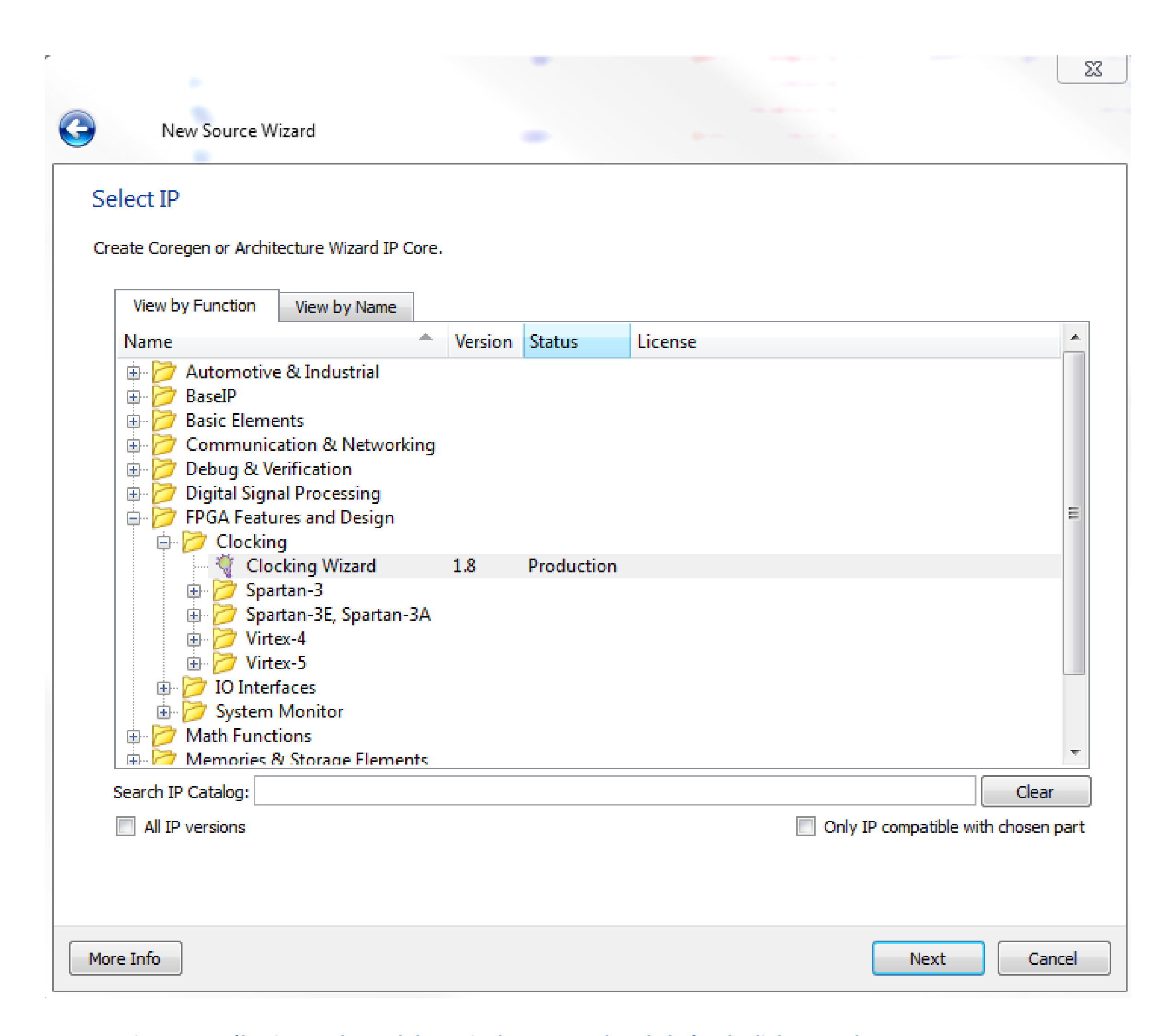
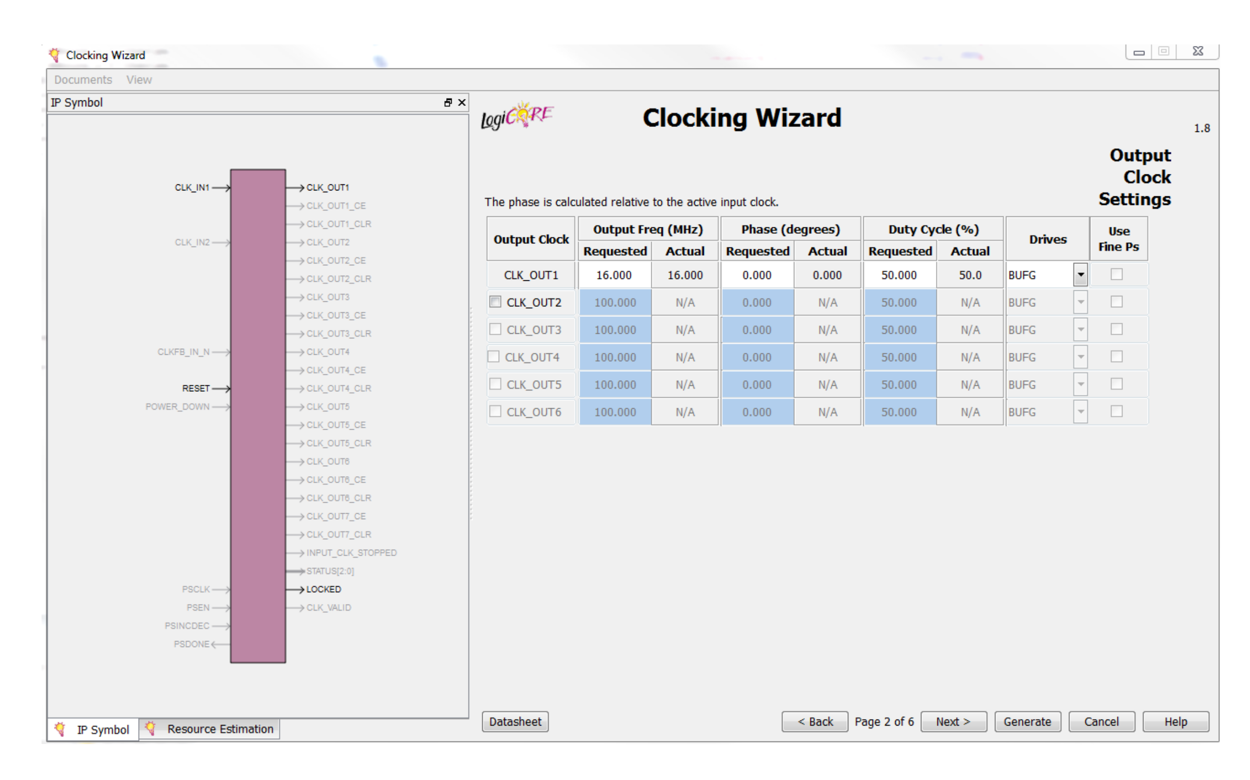



Figure 39. Sélectionnez le module IP Single DCM_IP dans la boîte de dialogue Select IP.

ISE lance alors le *Xilinx Clocking Wizard – General Setup*, illustré à la Figure 18. Dans cette boîte de dialogue tapez **Next**. Vous arrivez sur la deuxième fenêtre :

5. Dans la pageOutput Frequency Requested de CLK-OUT1, entrez **16**.
6. Cliquez sur **Next**.
7. Sur la page 3 il faut désélectionner **LOCKED**.
8. Sur la page 6 vous obtenez la liste des fichiers générés. Pour cela il suffit de cliquer sur **Generate**.

Le module dcm1 (fichier dcm1.xaw) est ajouté dans l'arborescence du projet de l'espace Hierarchy.




| Output Clock | Output Freq (MHz) | | Phase (degrees) | | Duty Cycle (%) | | Drives | Use Fine Ps |
|---|---|---|---|---|---|---|---|---|
| | Requested | Actual | Requested | Actual | Requested | Actual | | |
| CLK_OUT1 | 16.000 | 16.000 | 0.000 | 0.000 | 50.000 | 50.0 | BUFG | |
| CLK_OUT2 | 100.000 | N/A | 0.000 | N/A | 50.000 | N/A | BUFG | |
| CLK_OUT3 | 100.000 | N/A | 0.000 | N/A | 50.000 | N/A | BUFG | |
| CLK_OUT4 | 100.000 | N/A | 0.000 | N/A | 50.000 | N/A | BUFG | |
| CLK_OUT5 | 100.000 | N/A | 0.000 | N/A | 50.000 | N/A | BUFG | |
| CLK_OUT6 | 100.000 | N/A | 0.000 | N/A | 50.000 | N/A | BUFG | |

Figure 39. Le Clocking Wizard de ISE permet de générer et configurer des modules DCM sur mesure.

Le symbole représentant le module dcm1 est créé par ISE et placé dans la librairie locale de projet …./didact_sc/ipcore_dir.

## Création d'un module VHDL diviseur d'horloge

Le module d'anti-rebond et certains modules associés à la MSA du projet nécessitent des horloges moins rapides (sous les 10 kHz). Vous allez créer un diviseur d'horloge décrit en langage VHDL.

Pour créer le fichier source du diviseur :

1. Sélectionnez **Project > New Source.**
2. Sélectionnez **VHDL Module** dans la boîte de dialogue Select Source Type.
3. Entrez **diviseur_clk** comme nom de fichier et appuyer sur **Next**.

4. Créez un port d'entré **clkin** et trois ports de sortie **clk2hz, clk16hz** et **clk2khz** comme illustré à la Figure 40.
    - Entrez **clkin, clk2hz**, **clk16hz** et **clk2khz** dans les 4 premiers champs Port Name.
    - Dans le champ Direction, choisissez **in** pour clkin, et **out** pour clk2hz, clk16hz et clk2khz.
    - Ne cochez pas les champs Bus.

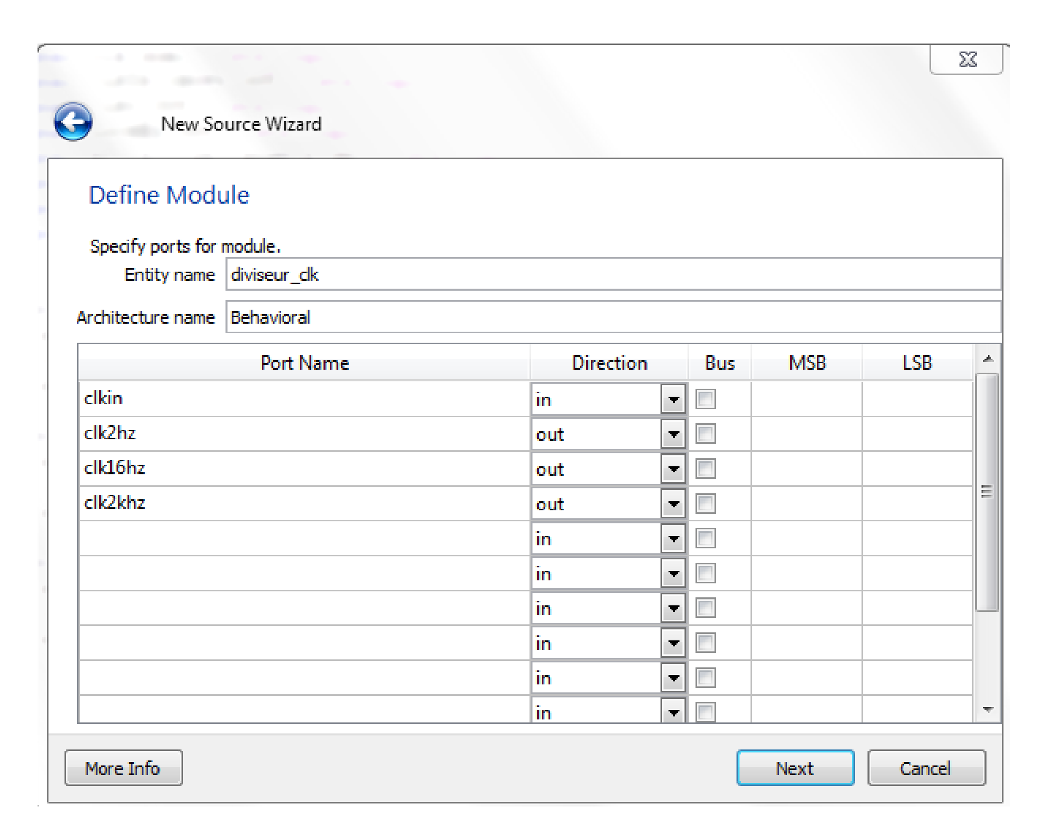



Figure 40. La boîte de dialogue Define Module pour entrer les ports d'entrées/sorties du module VHDL à créer.

5. Cliquez sur **Next** et ensuite sur **Finish** pour compléter cette étape de configuration.

Une boîte de dialogue affiche la description VHDL générée par le Wizard.

6. Cliquez sur **Finish**.

Le fichier VHDL vide décrivant le module diviseur_clk s'ouvre dans l'espace de travail comme montré à la Figure 41. Ce fichier contient l'entité du module mais présente une architecture vide à compléter.

### Complétez la description VHDL de diviseur_clk

Pour compléter la description VHDL du module diviseur_clk, effectuez les étapes suivantes :

1. Dans le fichier diviseur_clk ouvert dans l'espace de travail du Navigateur de Projet ISE, effacez **toutes** les lignes de code situées sous la ligne **`architecture Behavioral of diviseur_clk is.`**
2. Aller à l'Annexe 1 de ce document et visualisez la description VHDL originale du module HDL diviseur_clk. À l'aide de la souris, sélectionnez **toutes** les lignes de code situées sous la ligne **`architecture Behavioral of diviseur_clk is`** de cette description.
3. Copiez les lignes de code sélectionnées et collez-les dans le fichier diviseur_clk dans ISE, sous la ligne **`Behavioral of diviseur_clk is`**.

Le code VHDL du module du diviseur d'horloge est maintenant complété.

Révisez le tout au besoin pour éviter de provoquer des erreurs de syntaxe lors de la compilation du fichier.

**Note :** Ne portez pas attention aux lignes de code précédées des balises `--`. Ces balises désignent des lignes de commentaires. Elles n'ont donc aucune incidence sur le fonctionnement du module. Utilisez ces balises pour commenter votre code afin de vous y retrouver plus facilement.

4. Cliquez sur **File > Save**.
5. Sélectionnez l'item **diviseur_clk** dans l'espace Hierarchy et double-cliquez sur **Check Syntax** dans l'espace Processes.
6. Vérifiez les résultats dans la Console, corrigez les erreurs de syntaxe s'il y a lieu et fermer le fichier diviseur_clk.vhd.

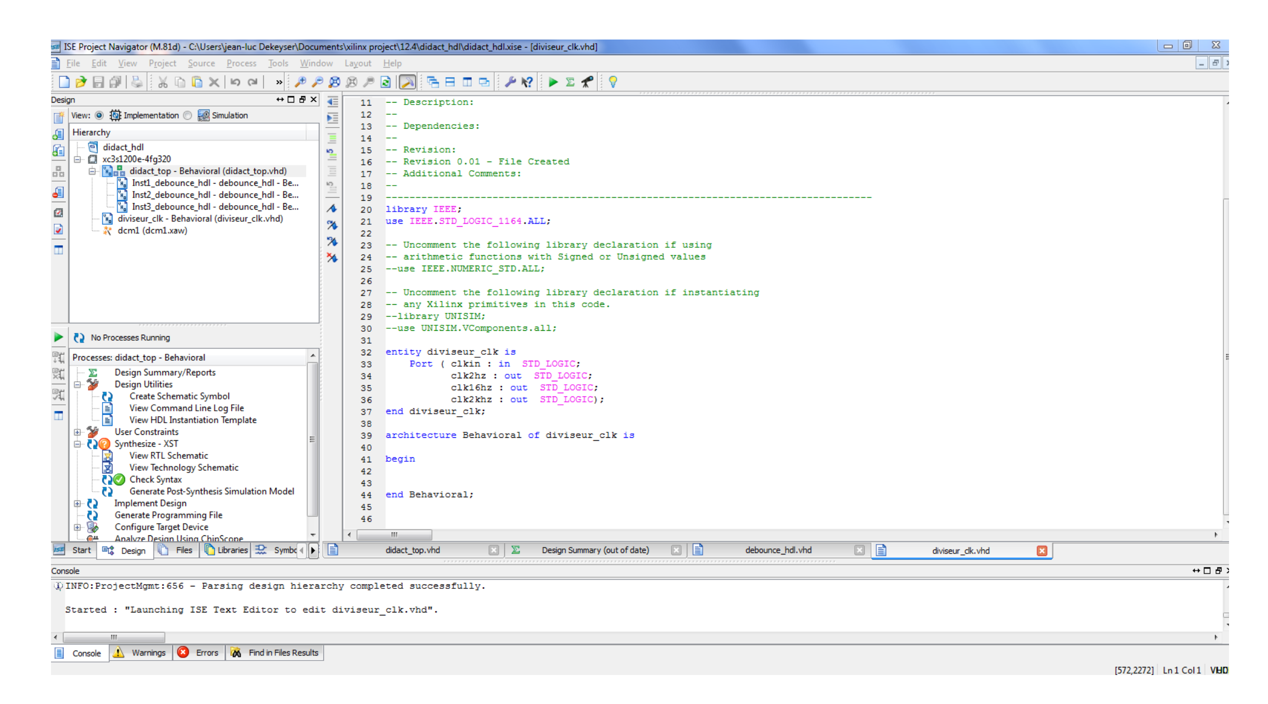


Figure 41. Fichier VHDL incomplet décrivant le module diviseur_clk.

Vous allez déclarer les composants dcm1 et diviseur_clk dans le fichier top_level et instancier chacun de ces sous modules.

## Déclarez et instanciez le module dcm1

Déclarez et instanciez le module dcm1 comme suit :

1. Dans l'espace Hierarchy du Navigateur de projet, sélectionnez l'item **dcm1 (dcm1.xaw)**.
2. Dans l'espace Processes (sous Hierachy), cliquez avec le bouton droit de la souris sur l'item **View HDL Instantiation Template** et sélectionnez **Process Properties**.
3. Assurez-vous que **VHDL** est sélectionné pour le champ HDL Instantiation Template Target Language value et cliquez sur **Ok**.
4. Dans l'espace Processes, double-cliquez sur **View HDL Instantiation Template.**
5. Dans la fenêtre de droite (fichier dcm1.vhi), sélectionnez le premier bloc code `component dcm1` … avec la souris, comme illustré à la Figure 42.

6. Copiez ce bloc de code à l'aide de la fonction **Edit > Copy** et collez-le dans votre fichier didact_top.vhd, juste sous le premier `end component` (sous la déclaration du component debounce_hdl , même si l'ordre importe peu, et avant les déclarations des signaux internes).

Référez-vous au fichier original didact_top.vhd placé à l'Annexe 6 de ce document pour vérifiez votre code.

```
-- The following code must appear in the VHDL architecture header:
------------- Begin Cut here for COMPONENT Declaration ------ COMP_TA
component dcm1
port
 (-- Clock in ports
  CLK_IN1           : in     std_logic;
  -- Clock out ports
  CLK_OUT1          : out    std_logic;
  -- Status and control signals
  RESET             : in     std_logic
 );
end component;

-- COMP_TAG_END ------ End COMPONENT Declaration ------------
-- The following code must appear in the VHDL architecture
-- body. Substitute your own instance name and net names.
------------- Begin Cut here for INSTANTIATION Template ----- INST_TA
your_instance_name : dcm1
  port map
   (-- Clock in ports
    CLK_IN1            => CLK_IN1,
    -- Clock out ports
    CLK_OUT1           => CLK_OUT1,
    -- Status and control signals
    RESET              => RESET);
-- INST_TAG_END ------ End INSTANTIATION Template ------------
```

Figure 42. Déclaration du composant dcm1 dans le HDL Instantiation Template. Le mapping des ports du composant avec les signaux interne est incomplet.

Instanciez ensuite ce composant :

1. Sélectionnez le fichier dcm1.vhi toujours ouvert dans l'espace de travail.
2. Cette fois, sélectionnez le tout le bloc `Inst_dcm1: dcm1 PORT MAP`… et copiez le.

3. Collez ce bloc dans le fichier top level, sous les trois blocs `inst1..3_debounce` et au dessus de end Behavioral.

Le mapping des ports du composant avec les signaux internes du module top level doit être complété avant de poursuivre.

4. Effectuez ce mapping :

```
Inst_dcm1: dcm1 PORT MAP(
      CLKIN_IN => clkin,
      RESET => rst,
      CLK_OUT1 => clk_dcm1,
);
```

5. Sauvegardez le schéma top level en sélectionnant **Save All**.

Remarquez que inst_dcm1 apparait maintenant sous l'item didact_top dans l'arborescence du projet.

6. Sélectionnez l'item **didact_top** dans l'onglet **Design** à gauche du Navigateur de projet de l'arborescence du projet (dans l'espace Hierarchy).
7. Dans l'espace Processes (sous Hierarchy), déployez l'item **Synthesize – XST**.
8. Cliquez avec le bouton de droite sur **Check syntax** et sélectionnez **ReRun** dans le menu déroulant.
9. Corriger les erreurs de syntaxe affichées dans la Console s'il y a lieu et fermer le fichier didact_top.vhd ainsi que le fichier dcm1.vhi.

## Déclarez et instanciez le module diviseur_clk

1. Dans l'espace Hierarchy du Navigateur de projet, double-cliquez sur l'item **didact_top**. Le fichier du schéma top level s'ouvre dans l'espace de travail.
2. Déclarez ensuite le composant diviseur_clk, la méthode précédente fonctionne aussi:
   - Juste sous le bloc de déclaration `component dcm1`…, ajoutez ces lignes de code :

```
component diviseur_clk
port(
      clkin : in  STD_LOGIC;
      clk2hz : buffer  STD_LOGIC;
      clk16hz : buffer  STD_LOGIC;
      clk2khz : buffer  STD_LOGIC);
```

```
end component;
```

**Note :** Vous n'avez pas de signaux internes à déclarer pour interconnecter ce bloc car ils ont déjà été tous déclarés à la section précédente Déclarez et instanciez le module d'anti-rebond.

3. Instanciez et interconnectez le module diviseur_clk avec les signaux internes et les ports d'entrées/sorties du module top level comme suit :
   - Sous le bloc d'instanciation du module dcm1 `Inst_dcm1: dcm1 PORT MAP…`, copiez et ajoutez ce bloc `port map` pour le module dcm1:

```
inst_diviseur_clk: diviseur_clk port map(
   clkin => clk_dcm1,
   clk2hz => clk2hz,
   clk16hz => clk16hz,
   clk2khz => clk2khz
);
```

4. Sauvegardez le schéma top level en sélectionnant **Save All**.

Remarquez que inst_diviseur_clk apparait maintenant sous l'item didact_top dans l'arborescence du projet.

5. Sélectionnez l'item **didact_top** dans l'onglet **Design** à gauche du Navigateur de projet de l'arborescence du projet (dans l'espace Hierarchy).
6. Dans l'espace Processes (sous Hierarchy), déployez l'item **Synthesize – XST**.
7. Cliquez avec le bouton de droite sur **Check syntax** et sélectionnez **ReRun** dans le menu déroulant.
8. Corriger les erreurs de syntaxe affichées dans la Console s'il y a lieu et fermer le fichier didact_top.vhd.

## Création d'une MSA à l'aide d'une description VHDL

Vous allez implémenter une MSA à l'aide d'une description VHDL et l'instancier dans le fichier top level avec les autres sous modules créés que vous interconnecterez entre eux pour réaliser le détecteur de séquence. Cette MSA utilise la même structure que celle représentée par le schéma de la Figure 6. Les variables d'état et les variables registrées (gardées en mémoire) seront affectées dans un processus registré (processus séquentiel mis à jour par un front d'horloge), alors que l'IFL et l'OFL de la MSA seront implémentés par des processus concurrents réalisant des circuits combinatoires.

Comme vous le verrez, l'approche par langage HDL est particulièrement adaptée à la réalisation de MSA de grandes complexités. En effet, le langage VHDL, par exemple, permet de transposer le diagramme d'état de la MSA directement dans une description comportementale (Behavioral) très intuitive.

Consultez la Figure 5 pour voir le diagramme d'état de la MSA du détecteur de séquence et le Tableau 1 pour une description de chaque état.

### Créez un nouveau fichier VHDL pour la MSA

Pour créer le fichier source de la MSA:

7. Sélectionnez **Project > New Source.**
8. Sélectionnez **VHDL Module** dans la boîte de dialogue Select Source Type.
9. Entrez **msa_hdl** comme nom de fichier et appuyer sur **Next**.
10. Créez cinq ports d'entrées **b0, b1, gs, clkin** et **rst**, et un port de sortie **enable_del** comme illustré à la Figure 43.
    - Entrez **b0, b1, gs, clkin, rst** et **enable_del** dans les 6 premiers champs Port Name.
    - Dans le champ Direction, choisissez **in** pour b0, b1, gs, clkin et rst, et **out** pour enable_del.
    - Ne cochez pas les champs Bus.
11. Cliquez sur **Next** et ensuite sur **Finish** pour compléter cette étape de configuration.

Une boîte de dialogue affiche la description VHDL générée par le Wizard.

12. Cliquez sur **Finish**.

Le fichier VHDL vide décrivant le module msa_hdl s'ouvre dans l'espace de travail. Comme pour les modules VHDL précédents, ce fichier contient l'entité du module mais présente une architecture vide à compléter.

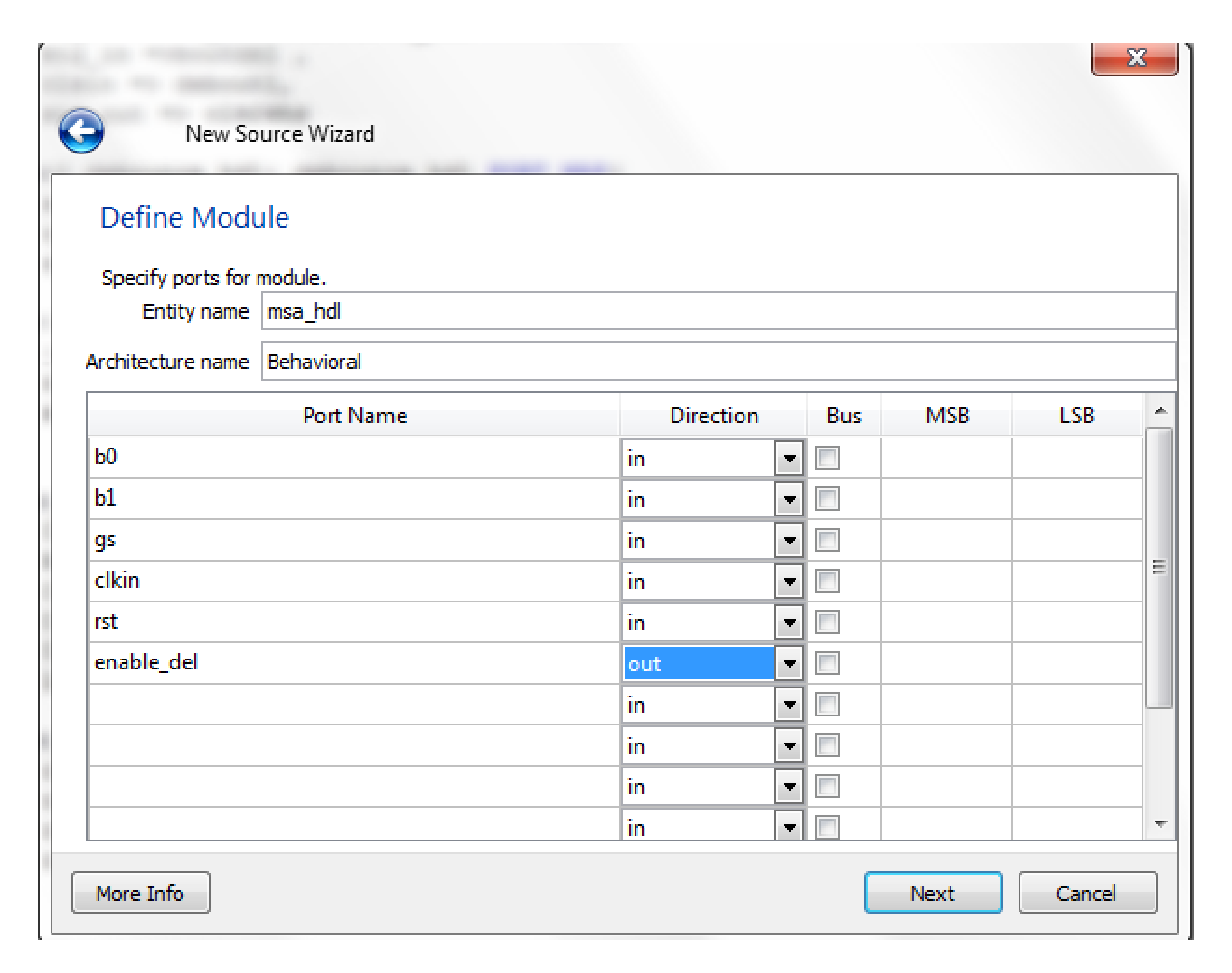



Figure 43. La boîte de dialogue Define Module pour entrer les ports d'entrées/sorties du module msa_hdl.

## Complétez la description VHDL de msa_hdl

Pour compléter la description VHDL du module msa_hdl, effectuer les étapes suivantes :

7. Dans le fichier msa_hdl ouvert dans l'espace de travail du Navigateur de Projet ISE, effacez **toutes** les lignes de code situées sous la ligne **`architecture Behavioral of msa_hdl is.`**
8. Aller à l'Annexe 5 de ce document et visualisez la description VHDL originale du module HDL msa_hdl. À l'aide de la souris, sélectionnez **toutes** les lignes de code situées sous la ligne **`architecture Behavioral of msa_hdl is`** de cette description.

9. Copiez les lignes de code sélectionnées et collez-les dans votre fichier msa_hdl dans ISE, sous la ligne **`Behavioral of msa_hdl is`**.

Révisez le tout au besoin pour évitez de provoquer des erreurs de syntaxe lors de la compilation du fichier.

Le code VHDL de la MSA est maintenant complété.

10. Cliquez sur **File > Save**.
11. Sélectionnez l'item **msa_hdl** dans l'espace Hierarchy et double-cliquez sur **Check Syntax** dans l'espace Processes.
12. Le résultat de la compilation s'affiche dans la Console. Corrigez les erreurs de syntaxe s'il y a lieu et fermez le fichier msa_hdl.vhd.

## Déclarez et instanciez le module msa_hdl

Procédez comme vous l'avez déjà fait pour les modules précédents afin de déclarer et instancier le module VHDL de la MSA :

1. Dans l'espace Hierarchy du Navigateur de projet, double-cliquez sur l'item **didact_top**. Le fichier du schéma top level s'ouvre dans l'espace de travail.
2. Déclarez ensuite le composant msa_hdl :
    - Juste sous le bloc de déclaration `component diviseur_clk…`, ajoutez ces lignes de code :

```
component msa_hdl
port(
      clkin : in  STD_LOGIC;
      rst : in  STD_LOGIC;
      b0 : in  STD_LOGIC;
      b1 : in  STD_LOGIC;
      gs : in  STD_LOGIC;
      enable_del : out  STD_LOGIC);
end component;
```

3. Instanciez le module msa_hdl et interconnectez-le avec les signaux internes et ports d'entrées/sorties du module top level comme suit :
    - Sous le bloc d'instanciation du module diviseur_clk `Inst_diviseur_clk: diviseur_clk PORT MAP…`, ajoutez ce bloc `port map` pour le module msa_hdl:

```
Inst_msa_hdl: msa_hdl port map(
     clkin => clk_dcm1,
     rst => rst,
     b0 => b0,
     b1 => b1,
     gs => gs,
     enable_del => enable_del
);
```

4. Sauvegardez le schéma top level en sélectionnant **Save All**.

Remarquez que msa_hdl apparait maintenant sous l'item didact_top dans l'arborescence du projet.

5. Sélectionnez l'item **didact_top** dans l'onglet **Design** à gauche du Navigateur de projet de l'arborescence du projet (dans l'espace Hierarchy).
6. Dans l'espace Processes (sous Hierarchy), déployez l'item **Synthesize – XST**.
7. Cliquez avec le bouton de droite sur **Check syntax** et sélectionnez **ReRun** dans le menu déroulant.
8. Corriger les erreurs de syntaxe affichées dans la Console s'il y a lieu et fermer le fichier didact_top.vhd.

## Complétez la description VHDL du module top level

Complétez le module top level didact_top en instanciant tous les sous modules requis et en réalisant toutes les connexions entre eux. Chaque module que vous avez créé dans les sections précédentes doit être déclaré à l'aide de la balise `component is` et instancié à l'aide de la balise `port map` dans l'architecture du module top level. Référez-vous au fichier original didact_top.vhd placé à l'Annexe 6 pour vérifier votre code du module top level.

Vous ajouterez les modules VHDL de l'encodeur, du générateur de signal gs (get something) et le registre à décalage à même le fichier top level. Pour ce faire, vous implémenterez ces modules dans des process que vous décrirez dans didact_top.

Effectuez les étapes suivantes pour compléter la description VHDL top level du détecteur de séquence synchrone :

1. Double-cliquez sur l'item **didact_top** dans l'arborescence du projet situé dans l'onglet Design.

Le fichier top level didact_top.vhd s'ouvre dans l'espace de travail.

2. Ensuite, ajoutez le code de l'encodeur comme suit :
   - Copiez le bloc de code suivant et collez-le juste sous le bloc d'instanciation du module msa_hdl `Inst_msa_hdl: msa_hdl port map(…,` dans le fichier top level :

```
-- Description de l'encodeur
b0 <= '1' when (debout3 = '0' and debout2 = '1' and debout1 = '0') or
                (debout3 = '1' and debout2 = '1' and debout1 = '0') else
          '0';
b1 <= '1' when debout3 = '1' and debout2 = '0' and debout1 = '0' else
          '0';
```

3. Sous ce dernier bloc, vous allez insérer le générateur de signal gs (get something).
   - Copiez le bloc de code suivant et collez-le sous la description de l'encodeur :

```
-- Description du Get something
gs <= '1' when debout3 = '1' or debout2 = '1' or debout1 = '1' else
          '0';
```

**Note :** Étudiez les descriptions VHDL de ces deux derniers modules. Notez qu'elles décrivent des circuits combinatoires et que leurs exécutions seront concourantes avec celles des process et autres composants déclarés dans l'architecture.

4. Copiez enfin la description du registre à décalage et collez-la sous la description du module get something, à la fin du fichier top level, juste au dessus du `end Behavioral;`.
   - Le code suivant décrit le registre à décalage :

```
-- Description du registre à décalage
xshifreg: process(rst,clk16hz)
begin
   if(rst = '1')then
          shreg <= (others => '0');
   elsif(clk16hz'event and clk16hz = '1')then
          if(enable_del = '1')then
                 shreg(0)<= clk2hz;
                 shreg(7 downto 1) <= shreg(6 downto 0);
          end if;
   end if;
end process;

Q_del <= shreg;
```

**Note :** Étudiez la description VHDL de ce dernier module. Notez qu'elle décrit bien un registre grâce à l'utilisation de la ligne de code `elsif(clk16hz'event and clk16hz = '1')then`. En effet, tous les signaux qui sont affectés sous cette balise seront reconnus comme des registres par les outils de synthèse et d'implémentation FPGA.

Le code VHDL du module top level est maintenant complété. Révisez votre fichier top level au besoin pour éviter de provoquer des erreurs de syntaxe lors de la compilation.

5. Cliquez sur **File > Save**.
6. Sélectionnez l'item **didact_top** dans l'onglet **Design** à gauche du Navigateur de projet de l'arborescence du projet (dans l'espace Hierarchy).
7. Dans l'espace Processes (sous Hierarchy), déployez l'item **Synthesize – XST**.
8. Cliquez avec le bouton de droite sur **Check syntax** et sélectionnez **ReRun** dans le menu déroulant.
9. Corriger les erreurs de syntaxe affichées dans la Console s'il y a lieu.

Fermez le fichier VHDL top level. Vous avez complété la conception du détecteur de séquence avec l'approche par langage HDL.

Pour poursuivre avec les étapes subséquentes de l'approche de conception par langage HDL :

- Allez au chapitre Simulation fonctionnelle avec ISim pour effectuer une simulation pré-synthèse du système.
- Ou allez au chapitre Implémentation du projet pour effectuer le placement routage du projet.

## Simulation fonctionnelle avec Isim

Une fois l'étape de conception terminée, vous allez effectuer une simulation fonctionnelle du projet à l'aide d'un testbench et du simulateur Isim. Pour ce faire, vous allez d'abord générer un fichier testbench VHDL à l'aide du Navigateur de projet ISE. Ensuite, vous configurerez et lancerez votre simulation Isim à partir du Navigateur ISE. Une fois appelé par ISE, Isim prendra en charge la simulation et la présentation des résultats.

**Note :** Nous assumons à cette étape que vous avez complétée l'étape préalable de conception du détecteur de séquence synchrone à l'aide d'une des deux approches de conception détaillées précédemment, soit l'approche de conception par schéma ou l'approche de conception par langage HDL.

### Étapes de configurations préalables

Ouvrez tout d'abord le projet du détecteur de séquence dans le Navigateur ISE, si ce n'est pas déjà fait. Vérifiez ensuite que ISE pointe correctement sur l'outil de simulation Isim. Suivez ces étapes :

1. Sélectionnez **Project > Design Properties**.
2. Vérifiez que Isim(VHDL/Verilog) est bien le Simulator
3. Cliquez sur **Ok** pour fermer la boîte de dialogue.

### Création d'un Testbench

Un testbench est un fichier VHDL non synthétisable utilisé comme cadre de niveau hiérarchique supérieur. Il permet d'instancier des modules VHDL à tester, de les connecter entre eux et de leurs assigner des valeurs de signaux de test. Le fichier testbench peut être simulé avec un outil de simulation HDL comme Isim ou Modelsim. Dans la section suivante, vous ajouterez un testbench au projet et vous le simulerez avec ISim pour tester le détecteur de séquence.

Ajoutez un fichier testbench VHDL au projet de la façon suivante :

1. Sélectionnez **Project > New Source.**
2. Sélectionnez **VHDL Test Bench** dans la boîte de dialogue Select Source Type.
3. Entrez **didact_top_tb** comme nom de fichier et appuyer sur **Next**.
4. Dans la boîte dialogue *Associate Source*, sélectionnez **didact_top** pour associer le fichier testbench didact_top_tb au fichier top level didact_top à simuler.

5. Cliquez sur **Next** et ensuite sur **Finish** pour créer le fichier testbench.

Le fichier VHDL du testbench créé s'affiche dans l'espace de travail, comme montré à la Figure 44. Observez la structure du fichier testbench. Contrairement aux modules VHDL courants, le testbench contient une section entity vide car il se trouve au niveau hiérarchique suprême et qu'il ne peut donc pas être instancié par un autre module VHDL.

Ce fichier sert essentiellement à instancier le module top level didact_top à l'aide des balises *component* et *port map* que vous verrez en vous déplaçant dans le fichier.

Note : Remarquez que le *port map* est précédé de l'identificateur UUT. Vous retrouverez cet identificateur lors de la simulation du testbench avec Isim.

Le testbench ne contient encore aucune assignation de signaux de test. Vous allez ajouter des assignations de signaux entre les balises prévues à cet effet dans le fichier :

```
-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN

     Ajoutez vos assignassions de signaux de test ici

      WAIT; -- will wait forever
   END PROCESS;
-- *** End Test Bench - User Defined Section ***
```

```vhdl
--
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY didact_top_didact_top_sch_tb IS
END didact_top_didact_top_sch_tb;
ARCHITECTURE behavioral OF didact_top_didact_top_sch_tb IS

   COMPONENT didact_top
   PORT( rst    :  IN STD_LOGIC;
          clkin    :  IN STD_LOGIC;
          bouton1 :  IN STD_LOGIC;
          bouton2 :  IN STD_LOGIC;
          bouton3 :  IN STD_LOGIC;
          Q_del    :  OUT    STD_LOGIC_VECTOR (7 DOWNTO 0));
   END COMPONENT;

   SIGNAL rst   :  STD_LOGIC;
   SIGNAL clkin    :  STD_LOGIC;
   SIGNAL bouton1 :  STD_LOGIC;
   SIGNAL bouton2 :  STD_LOGIC;
   SIGNAL bouton3 :  STD_LOGIC;
   SIGNAL Q_del    :  STD_LOGIC_VECTOR (7 DOWNTO 0);

BEGIN

   UUT: didact_top PORT MAP(
      rst => rst,
      clkin => clkin,
      bouton1 => bouton1,
      bouton2 => bouton2,
      bouton3 => bouton3,
      Q_del => Q_del
   );
```

Figure 44. Fichier testbench VHDL généré à l'aide du New Source Wizard.

La version complétée du fichier testbench se trouve à l'Annexe 7 de ce document. Effectuer les étapes suivantes pour compléter votre fichier didact_top_tb.vhd :

1. Dans fichier didact_top_tb.vhd complété reproduit à l'Annexe 7, sélectionnez, à l'aide de la souris, toutes les lignes de code situées entre les balises

```
"-- *** Test Bench - User Defined Section ***"
```
et
```
-- *** End Test Bench - User Defined Section ***
```

2. Copiez ces lignes de code et collez-les dans le fichier didact_top_tb, ouvert dans l'espace de travail du Navigateur de Projet ISE, entre les balises du même nom (Figure 45) :
   - Remplacez la section User Defined de votre fichier avec celle du fichier complété que vous avez copié en Annexe 7).
   - **Assurez-vous de donner une valeur initiale au signal d'horloge, soit d'avoir la ligne :** `SIGNAL clkin : STD_LOGIC := '0';`

Vérifiez le tout au besoin pour éviter de provoquer des erreurs de syntaxe lors du collage des lignes de code. Le testbench VHDL du détecteur de séquence est maintenant complété et prêt à être simulé.

3. Cliquez sur **File > Save** et fermez le fichier.

```
    );

-- *** Test Bench - User Defined Section ***
    tb : PROCESS
    BEGIN
       WAIT; -- will wait forever
    END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

Figure 45. Complétez le testbench avec les lignes de code disponibles à l'Annexe 7. Remplacez la section User Defined de votre fichier avec celle du fichier complété reproduit en Annexe 7.

## Simulation du projet avec Isim

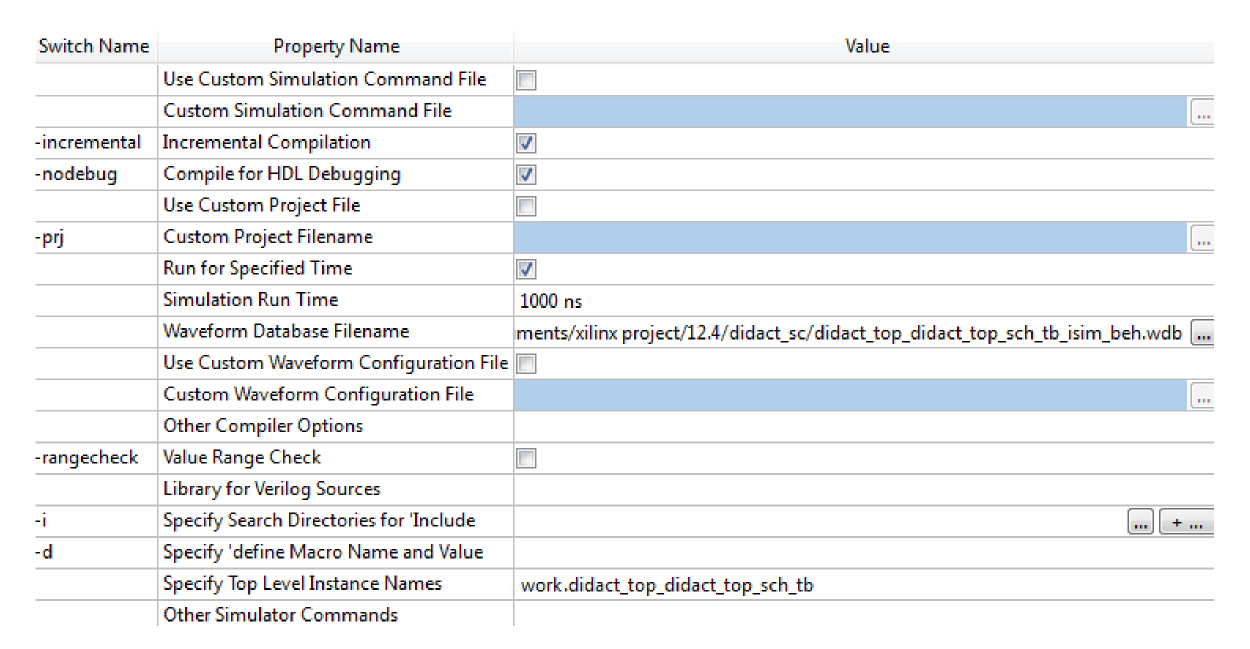
Vous allez lancer la simulation du testbench dans Isim à partir du Navigateur ISE. Configurez tout d'abord la simulation en effectuant les étapes suivantes :

1. Sélectionnez l'onglet **Design** dans l'espace Hierarchy situé dans la partie supérieure gauche du Navigateur de Projet.

2. Sélectionnez l'item **Behavioral** après avoir sélectionné **Simulation** juste au-dessus.




3. Sélectionnez le fichier top level **didact_top_tb** dans l'arborescence de projet.
4. Déployez l'item **ISim Simulator** dans la liste de l'espace Processes.
5. Cliquez avec le bouton droit de la souris sur **Simulate Behavioral Model**.
6. Choisissez l'onglet **Process Properties** du menu déroulant.
7. Les valeurs par défaut sont suffisantes
8. Cliquez sur **Ok**.



| Switch Name | Property Name | Value |
|---|---|---|
| | Use Custom Simulation Command File | ☐ |
| | Custom Simulation Command File | |
| -incremental | Incremental Compilation | ☑ |
| -nodebug | Compile for HDL Debugging | ☑ |
| | Use Custom Project File | ☐ |
| -prj | Custom Project Filename | |
| | Run for Specified Time | ☑ |
| | Simulation Run Time | 1000 ns |
| | Waveform Database Filename | ments/xilinx project/12.4/didact_sc/didact_top_didact_top_sch_tb_isim_beh.wdb |
| | Use Custom Waveform Configuration File | ☐ |
| | Custom Waveform Configuration File | |
| | Other Compiler Options | |
| -rangecheck | Value Range Check | ☐ |
| | Library for Verilog Sources | |
| -i | Specify Search Directories for 'Include | |
| -d | Specify 'define Macro Name and Value | |
| | Specify Top Level Instance Names | work.didact_top_didact_top_sch_tb |
| | Other Simulator Commands | |

Figure 46. Boîte de dialogue des options de simulation dans ISim.

Assurez-vous que votre arborescence de projet est conforme à celle montrée à la Figure 47 et lancez la simulation en double-cliquant sur **Simulate Behavioral Model** avec le bouton de gauche de la souris.

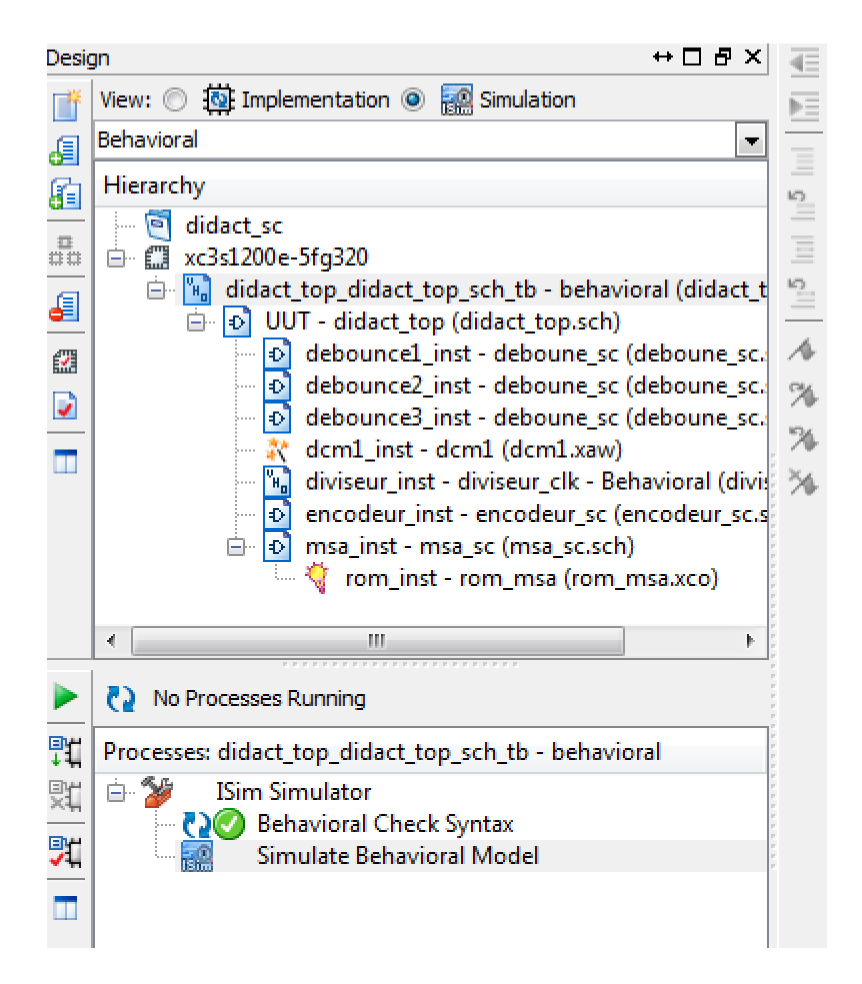



**Figure 47. Onglet Design en mode Source for : Behavioral Simulation pour lancer la simulation fonctionnelle.**

Le logiciel ISim s'ouvre comme illustré à la Figure 48.

Remarquez à gauche, dans l'onglet Instances and Processes, on retrouve l'item UUT sous didact_top_didact_top_sch_tb. L'item UUT (qui signifie unit under test) désigne le module top level instancié dans le fichier testbench.

De plus, en déployant l'item UUT on note que tous les sous modules instanciés dans le fichier VHDL top level sont répertoriés sous cet item.

Les résultats de simulation seront affichés dans la fenêtre Default.wcfg , à droite de la fenêtre ISim.

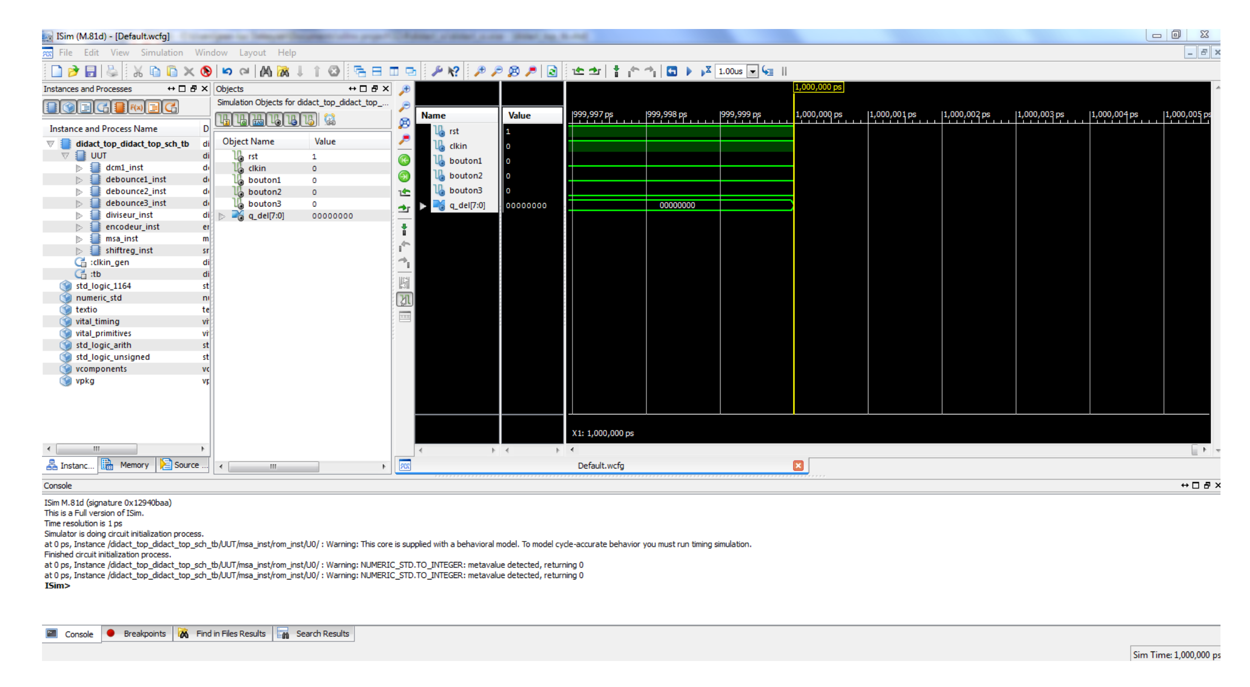



Figure 48. La fenêtre de simulation ISim s'ouvre.

## Configurez la simulation dans ISim

1. En cliquant sur un sous-modules, par exemple UUT, on voit apparaître dans la fenêtre *Simulation objects* la liste des identificateurs de signaux . On peut ainsi les faire glisser dans la fenêtre de simulation en dessous de q_del. Faites glisser etatpres, etatsuiv, enable_del puis rejouez la simulation.
2. En jouant sur le zoom et en déplaçant la ligne jaune on peut observer les valeurs à chaque instant de la simulation. Vérifiez les changements d'état qui doivent correspondre à notre MSA.

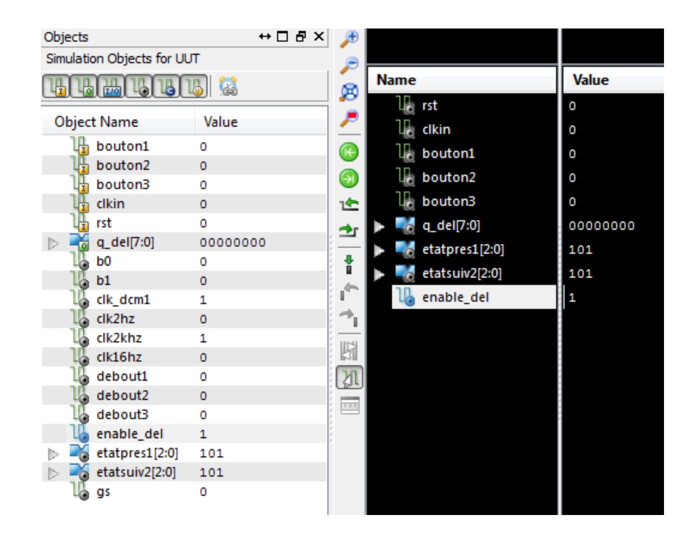



**Figure 49. Ajoutez les signaux d'entrées/sorties du module UUT à la Fenêtre Wave pour visualiser leur comportement après la simulation Simulation des signaux de la MSA avec ISim.**

Configurez ensuite le temps de simulation :

- Entrez **30 ms** dans le Run Length dans la barre d'outils, comme montré à la Figure 50.




Figure 50. Ajustez le temps de simulation et lancer la simulation en cliquant sur la flèche droite

## Lancez la simulation

Enfin, lancer la simulation en cliquant sur l'icône **Run for the time specified** dans la barre d'outils (Figure 50).

La simulation s'exécute en quelques secondes et les résultats s'affichent dans la fenêtre Wave.

## Visualisez les résultats

Pour vérifier les résultats de simulation, cliquez d'abord sur l'icône **Zoom to Full View** de la barre d'outils.

La valeur des signaux s'affiche dans la fenêtre Wave pour la totalité du temps de simulation.



**Figure 51. Cliquez sur l'icône Zoom to Full View pour visualiser la simulation complète sur 30 ms.**

Finalement, vérifiez que vos résultats de simulation sont conformes à ceux montrés à la Figure 52. Vérifiez que la MSA passe par la bonne séquence d'états en regardant les signaux etatpres et etatsuiv. Remarquez que le signal enable_del est activé avant la fin de la simulation. Pour observer la variation des signaux q_del , il faut laisser la simulation se dérouler plusieurs centaines de ms.

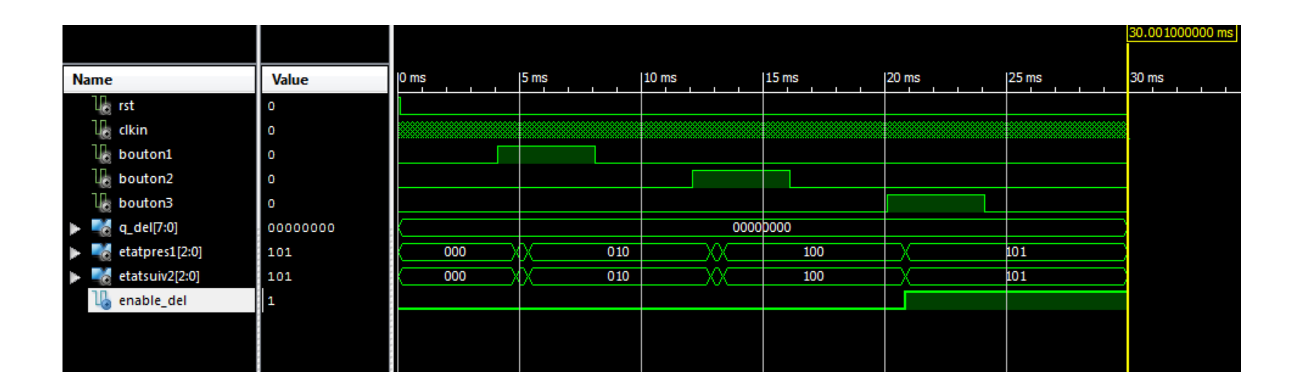



Figure 52. Les résultats de simulation s'affichent dans la fenêtre Wave pour la totalité du temps de simulation.

Fermez le logiciel ISim. Vous avez complété l'étape de simulation fonctionnelle du détecteur de séquence.

Vous êtes prêts à effectuer l'implémentation du projet :

- Allez au chapitre Implémentation du projet pour effectuer le placement routage du détecteur de séquence.

# Implémentation du projet

Dans ce chapitre, vous effectuerez la synthèse et l'implémentation du projet du détecteur de séquence synchrone. Vous créez d'abord des contraintes de timing et les incorporer au projet. Vous spécifierez ensuite la configuration des broches d'entrées/sorties à adopter et vous générerez enfin le fichier de programmation du FPGA Spartan 6 de la carte Nexys3.

L'implémentation du projet consiste à effectuer les étapes d'interprétation (translation), de mapping, de placement & routage et de génération du fichier de programmation BIT. Tous les outils d'implémentation du projet sont intégrés au Navigateur ISE. Pour plus d'information sur les étapes d'implémentation et sur chaque outil, consultez le ISE In-Depth Tutorial version 12.4 et le Manuel d'utilisation ISE version 12.4.

**Note :** Nous assumons à cette étape que vous avez complétée l'étape préalable de conception du détecteur de séquence synchrone à l'aide d'une des deux approches de conception détaillées précédemment, soit l'approche de conception par schéma ou l'approche de conception par langage HDL.

## Choix des options

Vous devez tout d'abord configurer les propriétés des outils d'implémentation. Suivez ces étapes :

1. Sélectionnez l'onglet **Design** dans l'espace Hierarchy situé dans la partie supérieure gauche du Navigateur de Projet.
2. Assurez-vous que l'item **Implementation** est sélectionné dans la liste **View** au dessus de l'espace Hierarchy.
3. Sélectionnez l'item top level **didact_top** dans l'arborescence de projet.
4. Dans l'espace Processes, sous l'espace Hierarchy, cliquez avec le bouton de droite de la souris sur l'item **Implement Design**. Selectionnez Process Properties.
5. La boîte de dialogue Process Properties apparaît. Sélectionnez l'item **Place & Route Properties** de la liste Category de Gauche.
6. Assurez-vous que l'item **Advanced** est sélectionné dans la liste Property display level, dans le bas de la boîte de dialogue.
7. Changez le Place & Route Effort Level (Overall) pour **High** dans la colonne Value juste à côté.

Vous devriez avoir un résultat conforme à la Figure 53.

8. Cliquez sur **Ok** pour fermer la boîte de dialogue.

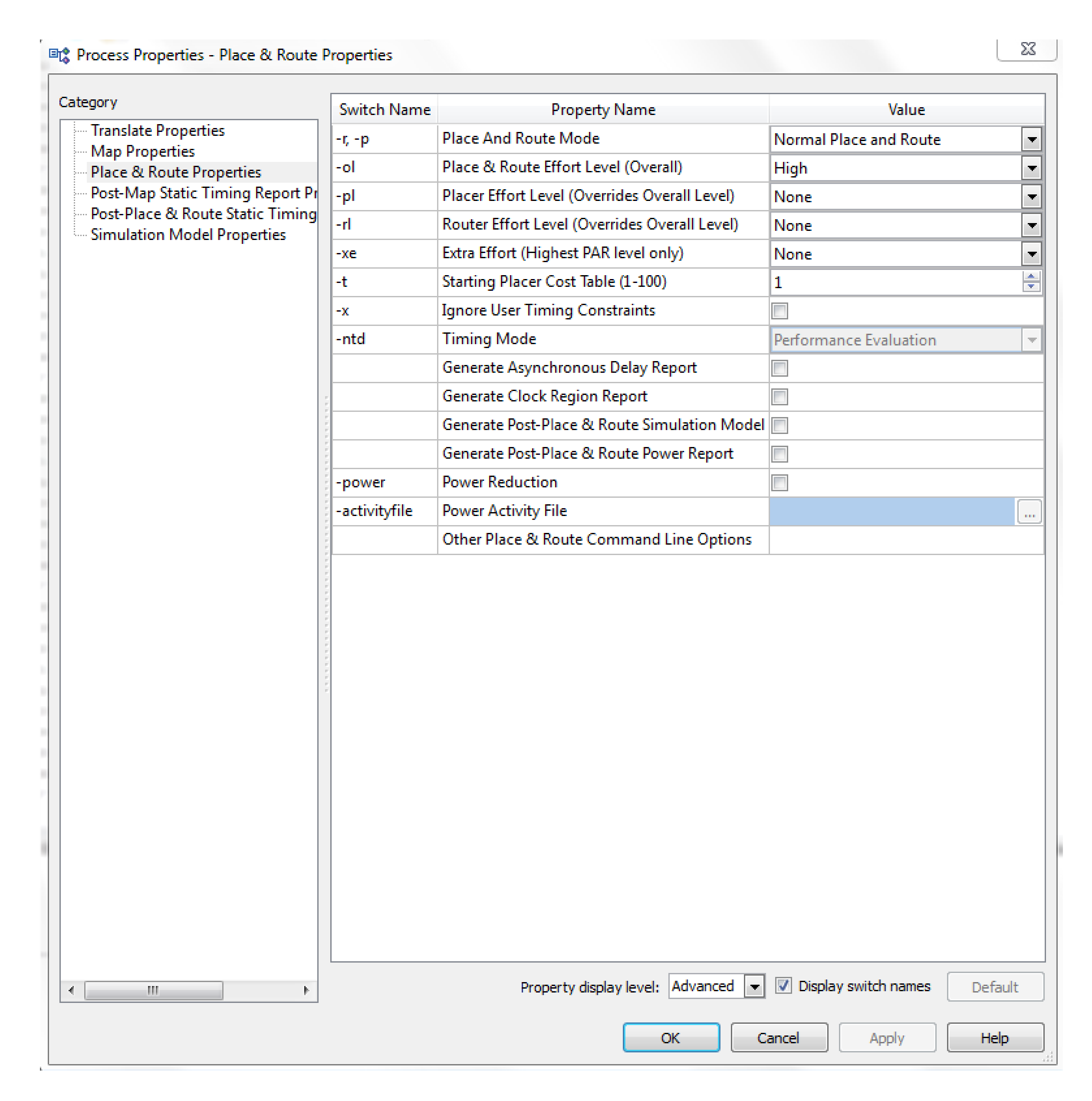



Figure 53. Configuration à adopter pour la boîte de dialogue Process Properties pour les options de placement et routage.

## Création de contraintes de timing

Les contraintes de timings du design sont entrées dans un fichier UCF (User constraints file). ISE met à votre disposition des outils graphiques comme le Constraints Editor et PlanAhead pour éditer ce fichier automatiquement.

Vous allez, tout d'abord, entrer des contraintes de timing à l'aide du Constraints Editor. Ces contraintes seront inscrites automatiquement dans un fichier UCF qui sera associé au projet en cours.

Pour lancer le Constraints Editor :

1. Sélectionnez **didact_top** dans l'arborescence du projet.
2. Déployez l'item **User Constraints** dans la liste de l'espace Processes.
3. Double-cliquez sur l'item **Create Timing Constraints**.
4. Cliquez **Yes** sur la boîte de dialogue demandant si vous voulez créer un fichier UCF.

Remarquez que l'étape de synthèse du projet s'effectue automatiquement dans l'espace Processes.

ISE ouvre ensuite le Constraints Editor comme montré à la Figure 54. Les trois horloges utilisées dans le détecteur sont répertoriées dans la liste Unconstrained Clocks.

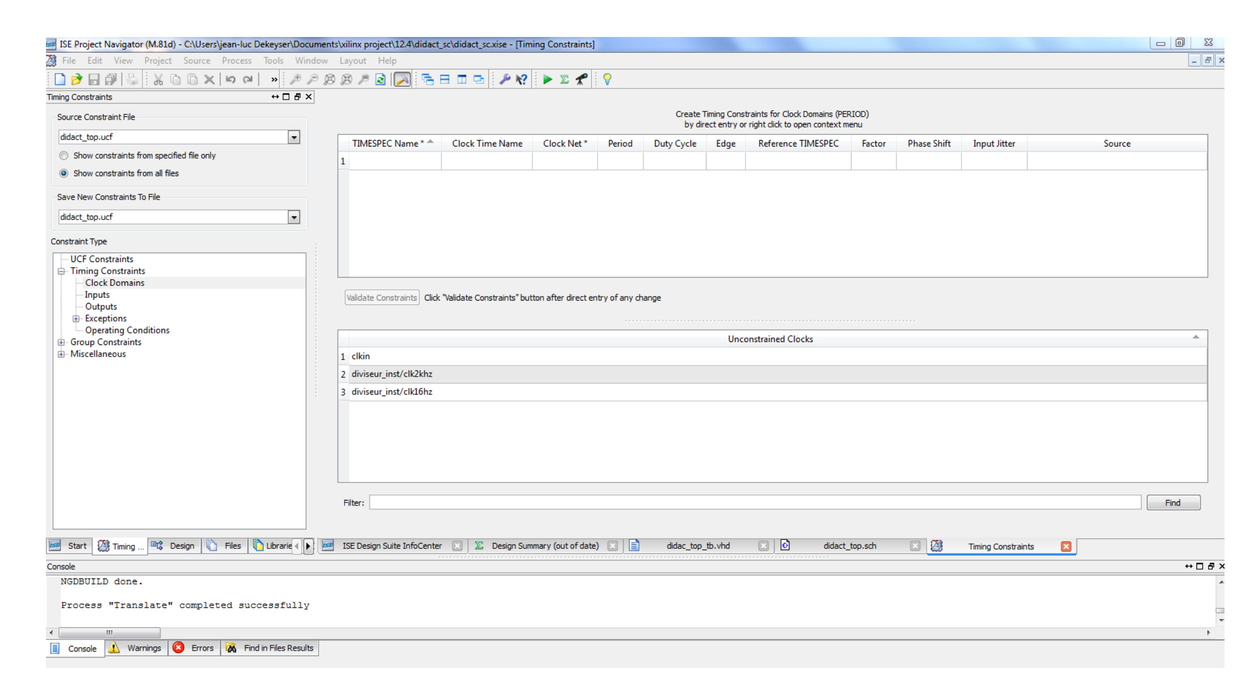


Figure 54. Le Constraints Editor vous permet d'entrer des contraintes de timing sur les horloges et les signaux d'entrées/sorties.

Vous allez spécifier des contraintes de timing pour les trois horloges détectées. Commencez par spécifier l'horloge de référence du système en procédant comme suit :

1. Assurez-vous que l'item **Clocks Domains** est sélectionné dans l'espace Constraint Type.
2. Double-cliquez sur l'item **clkin** de la liste Unconstrained Clocks au centre du Navigateur de projet.

La boîte de dialogue Clock Period s'ouvre. Effectuez les configurations suivantes :

1. Assurez-vous que Specify time est sélectionné et inscrivez **10** dans le champ Time juste dessous. Assurez-vous que **ns** est sélectionné dans la liste Units.
2. Inscrivez **50** dans le champ Rising duty cycle et assurez-vous que le champ Units associé soit à **%**.
3. Assurez-vous que les autres items sont configurés conformément à la boîte de dialogue Clock Period montrée à la Figure 55.
4. Cliquez sur **Ok**.

Vous constaterez que la nouvelle contrainte nommée TS_clkin est ajoutée dans la liste du haut.

Sauvegardez le projet et fermer le Constraints Editor.

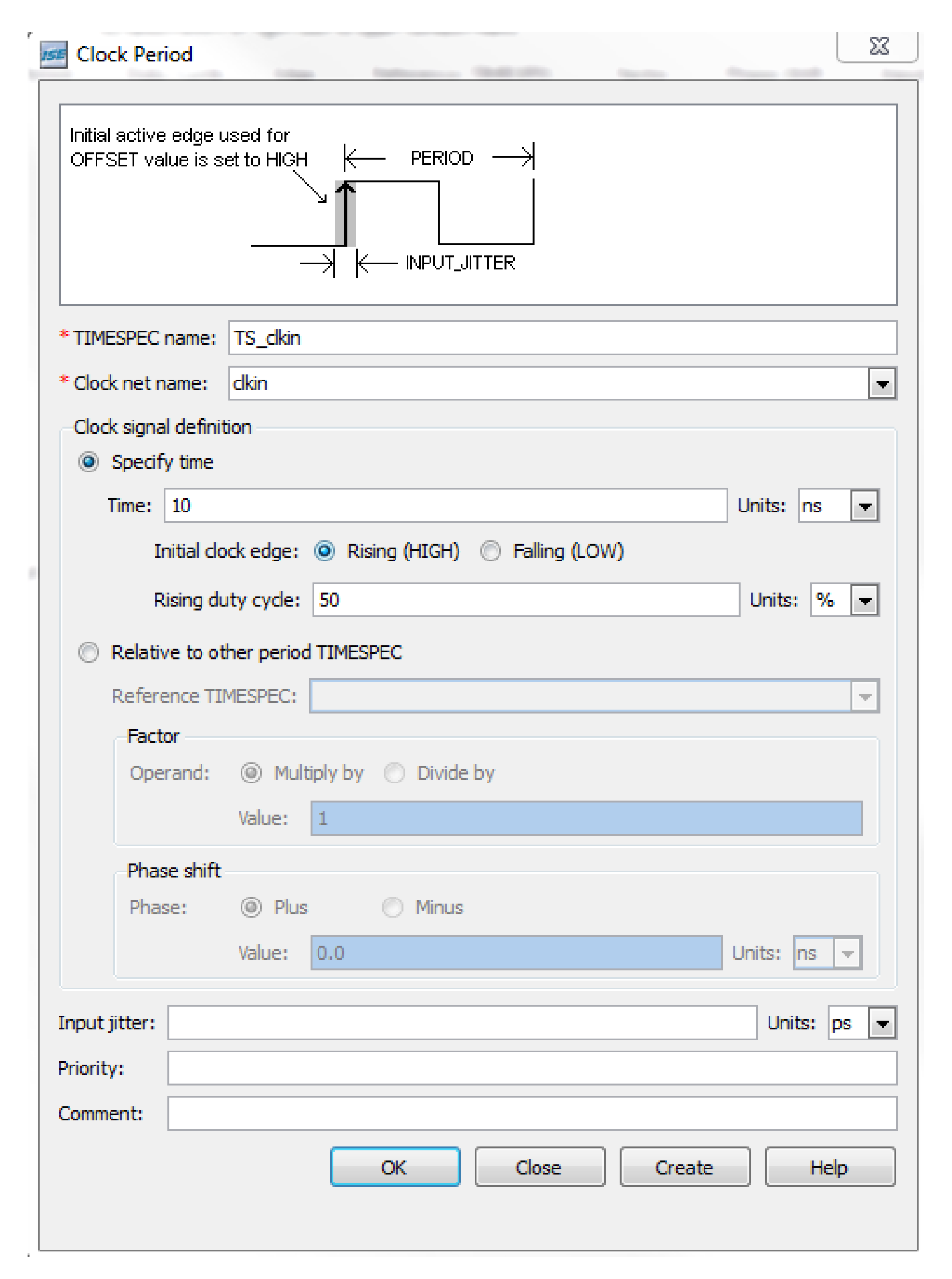



**Figure 55. Boîte de dialogue Clock Period pour ajouter des contraintes de timing sur les horloges utilises dans le détecteur de séquence.**

## Assignation des broches du FPGA avec PlanAhead

L'outil intégré PlanAhead est utilisé pour associer les broches du FPGA et pour créer des contraintes individuelles ou de groupe sur les caractéristiques physiques et électriques des entrées/sorties.

1. Sélectionnez **didact_top** dans l'arborescence du projet.
2. Déployez l'item **User Constraints** dans la liste de l'espace Processes.
3. Double-cliquez sur l'item **I/O Pin Planning (PlanAhead) –Post-Synthesis**.

Une boîte de dialogue s'ouvre et vous offre de consulter la documentation appropriée pour en savoir plus sur PlanAhead. Fermer cette boîte de dialogue pour l'instant.

L'outil PlanAhead s'ouvre.

Spécifiez les broches du FPGA comme suit :

1. Cliquez d'abord sur le contour noir de la fenêtre package et cliquez ensuite deux fois sur l'icône Zoom in de la barre d'outils.
2. Dans la fenêtre/onglet I/O Ports, déployez la liste Scalar ports en cliquant sur le **+**.
3. Dans cette liste, cliquez sur **clkin**.
4. Dans la fenêtre Package Pins, déployez la liste **I/O Bank 2**.
5. Cliquez sur **Name** dans le haut de la colonne pour classer les éléments de la liste I/O Bank 2 en ordre croissant.

Vous allez assigner la broche V10 au port/étiquette clkin. Cela aura pour effet de connecter l'oscillateur à 100 MHz de la carte Nexys3 à l'entrée CLK_IN du module de gestion d'horloge dcm1 utilisé dans ce projet.

6. Sélectionnez **clkin** dans le **I/O port** et par un drag and drop venez le placer sur la case V10 du **package**.

Vous devriez obtenir un résultat conforme à celui montré à la Figure 56.

7. Positionnez ensuite le curseur sur la fenêtre Package et repérez la broche **B8** au centre de la partie supérieure du package. Notez que la broche B8 est mise en évidence pour vous aider à la localiser.
8. Positionnez le curseur sur la broche C9. Le curseur affiche ce texte « Place clkin at B8 ». La Figure 57 illustre cette étape.
9. Cliquez sur la broche **B8** pour l'assigner au signal **clkin**.

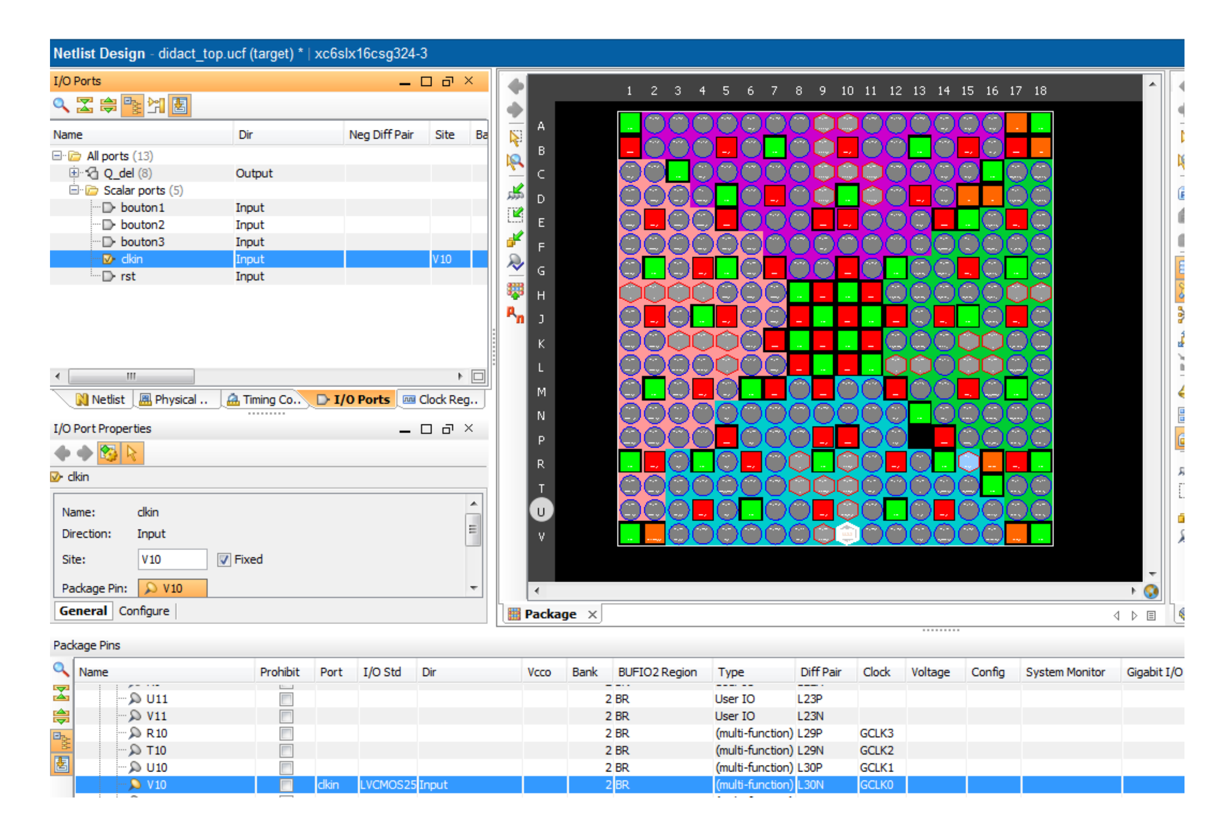



**Figure 56. L’outil PlanAhead permet d’assigner les broches du FPGA et de créer des contraintes de design sur les entrées/sorties. Cette figure montre comment assigner la broche V10 au port/étiquette clkin du design.**

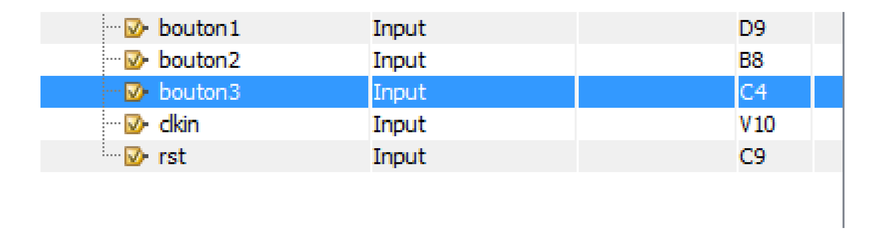



**Figure 57. Assigner les port/étiquette aux broches dans la fenêtre I/O ports.**

## Assignez les autres broches du FPGA

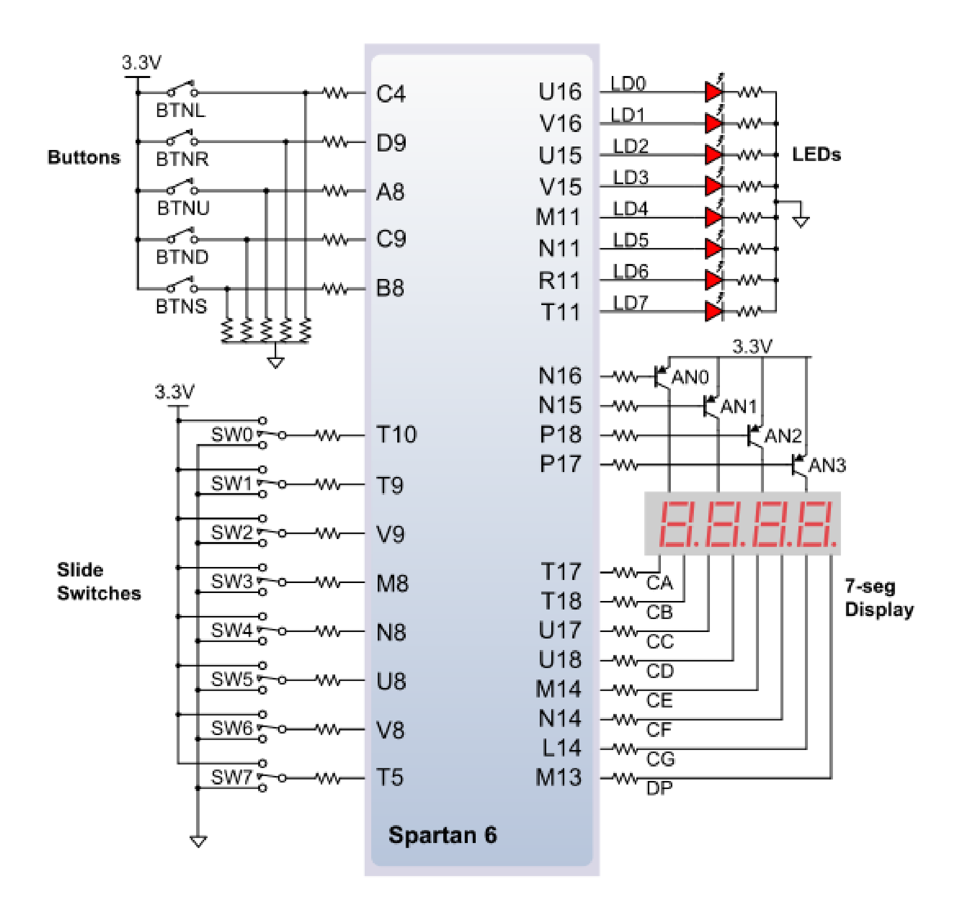
Les étapes suivantes vous guideront dans l’assignation des autres broches du FPGA. Vous allez assigner le signal de remise à 0 (rst) au bouton poussoir BTND de la carte, le signal bouton1 au bouton BTNR, le signal bouton2 au bouton BTNS et le bouton3 au bouton BTNL ( ordre de saisie du

code valide 1 2 3 , il faut changer l'assignation pour changer le code). Enfin, vous allez assigner les bits du bus Q_del(7:0) au Leds LD7 à LD0 de la carte. Procédez comme suit pour réaliser ces connexions :

1. Assignez le signal rst dans la fenêtre I/O Ports à la broche C9 comme vous l'avez fait précédemment pour assigner clkin.
2. Assignez le signal bouton1 dans la fenêtre I/O Ports à la broche D9.
3. Assignez le signal bouton2 dans la fenêtre I/O Ports à la broche B8.
4. Assignez le signal bouton3 dans la fenêtre I/O Ports à la broche C4.

Vous avez assigné tous les ports/étiquettes d'entrées aux broches désirées.

Assignez ensuite le port/étiquette Q_del(7:0) aux DEL LD7 à LD0. Procédez de la même façon broche par broche.




1. Sauvegardez votre assignation.

Vous avez assigné tous les ports/étiquettes aux broches de FPGA souhaitées. Consultez le Tableau 5 pour un résumé des assignations et des liens entre les port/étiquettes du projet et les composants physiques de la carte Nexys2.

Fermer PlanAhead et revenez au Navigateur de projet ISE.

Tableau 5. Résumé des assignations de broches et composants physiques visés.

| | | | |
|---|---|---|---|
| Q_del (8) | Output | | |
| Q_del[0] | Output | | U16 |
| Q_del[1] | Output | | V16 |
| Q_del[2] | Output | | U15 |
| Q_del[3] | Output | | V15 |
| Q_del[4] | Output | | M11 |
| Q_del[5] | Output | | N11 |
| Q_del[6] | Output | | R11 |
| Q_del[7] | Output | | T11 |
| Scalar ports (5) | | | |
| bouton1 | Input | | D9 |
| bouton2 | Input | | B8 |
| bouton3 | Input | | C4 |
| clkin | Input | | V10 |
| rst | Input | | C9 |

## Consultez le fichier UCF créé

Vous pouvez consultez le fichier UCF créé suite aux ajouts de contraintes de timings et d'assignations de broches. Pour ce faire :

1. Sélectionnez le fichier **difact_top.ucf** (…\didact_sc\didact_top\didact_top.ucf) dans l'arborescence de projet de l'espace Hierarchy.
2. Double-cliquez sur **Edit Constraints (Text)** dans la liste User Constraints.

Le fichier UCF s'ouvre dans l'espace de travail. Vous retrouverez les contraintes spécifiées sur les trois horloges et les assignations de broches annotées dans ce fichier. Familiarisez-vous avec la syntaxe de ce fichier. Vérifier que le contenu de votre fichier est conforme à celui du fichier original placé à l'Annexe 8.

On peut écrire directement ce fichier sans passer par PlanAhead, en particulier pour des projets utilisant les mêmes types de connexion.

## Implémentez le projet

L'implémentation du projet s'effectue en trois étapes, soit l'interprétation (translation), le mapping et le placement & routage. ISE appel automatiquement l'outil intégré approprié pour réaliser successivement chaque étape.

1. Sélectionnez **didact_top** dans l'arborescence du projet.
2. Double-cliquez sur l'item **Implementing Design**.
3. Consultez les messages dans la Console et assurez-vous que l'implémentation du projet s'est bien déroulée.

ISE lance automatiquement les étapes d'implémentation les unes après les autres. Nous rappelons que l'implémentation comporte les trois étapes suivantes : l'interprétation (translation), le mapping et le placement & routage.

Après qu'ISE ait terminé toutes les étapes, visualisez les résultats dans la Console.

## Consultez le rapport de synthèse et d'implémentation

Pour consulter le rapport de synthèse et d'implémentation du projet, cliquez sur l'icône **Design Summary/Reports** de la barre d'outils.



Figure 58. Cliquez sur cet icône pour consulter le rapport d'implémentation.

Le sommaire du rapport et une arborescence de fichiers s'ouvrent dans l'espace de travail. Le rapport est divisé en cinq sections.

Tableau 6. Les différentes sections du rapport de projet et leur utilité.

| Section | Description |
|---|---|
| Projet Status | Cette section du rapport donne l'heure de création ainsi que l'état du projet et son avancement. En regardant cette section, vous saurez à quelle étape vous vous situez dans le flow de conception, si le design comporte des erreurs ou des warnings et vous aurez divers détails sur l'étape en cours dont les résultats de placement & routage et le statut des contraintes de timings. |
| Device Utilization Summary | Cette section trace un portrait de l'utilisation des ressources dans le FPGA. On y dresse une liste des différents types de composants disponibles dans le FPGA ainsi que la quantité utilisée dans le projet pour chacun. |
| Performance Summary | Cette section donne accès à plusieurs rapports de performance portant sur les horloges, les contraintes de timings, la configuration des broches, etc. |
| Detailed Reports | Cette section donne accès à des rapports détaillés portant sur chaque étape de l'implémentation dont la synthèse, l'interprétation (Translation), le mapping, le placement & routage, les contraintes de timing et la génération du fichier de programmation. |
| Secondary Reports | Cette section donne de l'information supplémentaire sur certaines étapes du flow de conception dont la simulation avec timing. |

Cliquez sur les hyperliens de chaque section du rapport pour ouvrir les rapports détaillé portant sur les différents items.

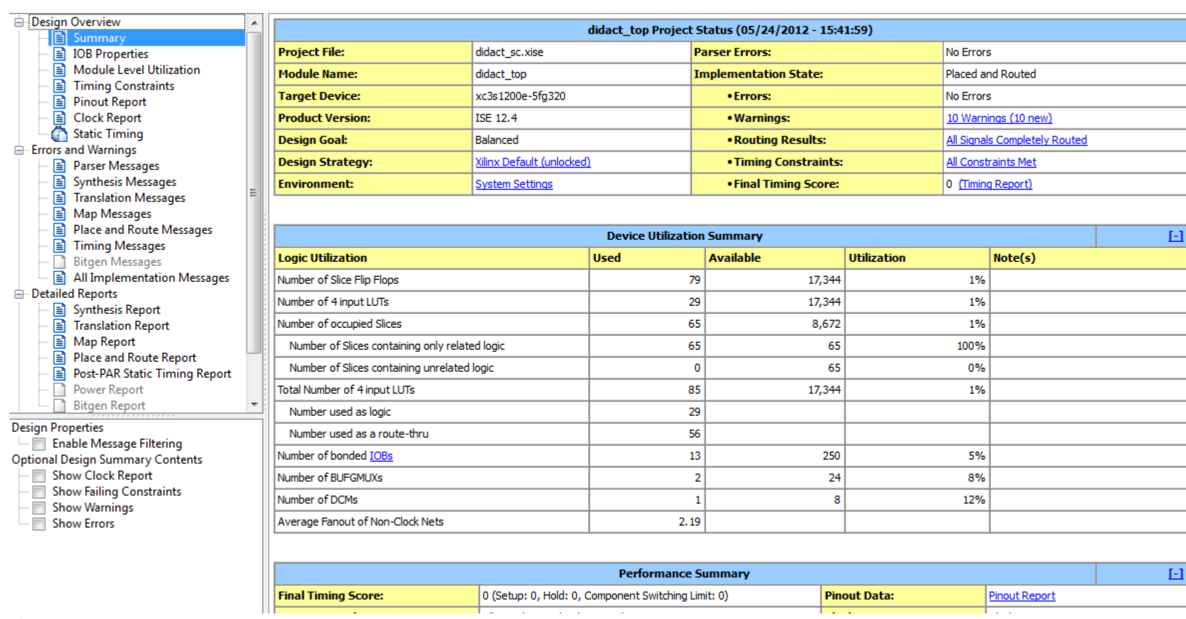


**Figure 59. Le rapport d'implémentation généré par le Navigateur de projet donne tous les résultats et les détails de l'implémentation du projet.**

Vous avez complété l'implémentation du détecteur de séquence.

Pour poursuivre avec les étapes subséquentes du flow de conception :

- Allez au chapitre Simulation avec timings pour effectuer une simulation avec délais réels.
- Allez au chapitre Configuration de la carte Nexys3 pour configurer la carte Nexys3 avec le projet du détecteur de séquence.

## Simulation avec timings

Une fois l'étape d'implémentation du projet terminée, vous allez effectuer une simulation avec timings. Vous réaliserez cette simulation à l'aide du testbench que vous avez créé à la section Simulation fonctionnelle avec ISim et du simulateur ISim. La simulation avec timings (ou post placement & routage) est une étape essentielle du flot de conception pour s'assurer que le fonctionnement du projet est conforme avant de le réaliser physiquement.

La simulation avec timings utilise les informations sur les délais dans les blocs FPGA et dans le routage afin de reproduire le comportement réel du projet avec grande précision. La simulation avec timing utilise l'information détaillée sur le placement & routage générée lors l'étape d'implémentation du projet. Une fois appelé par ISE, ISim intégrera l'information sur les timings et prendra en charge la simulation et la présentation des résultats.

**Note :** Nous assumons à cette étape que vous avez complétée l'étape préalable d'implémentation du projet détaillée précédemment dans ce didacticiel.

### Étapes de configurations préalables

Tout d'abord, assurez-vous d'avoir effectué toutes les étapes de configuration préalables décrites dans la section Simulation fonctionnelle. Ensuite, assurez-vous d'avoir créé un testbench et de l'avoir intégré au projet, comme montré dans l'étape Création d'un Testbench de cette même section.

### Simulation avec timing dans ISim

Vous allez lancer la simulation du testbench avec timing dans ISim à partir du Navigateur ISE. Configurez tout d'abord la simulation en effectuant les étapes suivantes :

1. Sélectionnez l'onglet **Design** dans l'espace Hierarchy situé dans la partie supérieure gauche du Navigateur de Projet.
2. Sélectionnez l'item **Post-Route** dans la liste juste au-dessous de View.
3. Sélectionnez le fichier top level **didact_top_tb** dans l'arborescence de projet.
4. Déployez l'item **ISim simulator** dans la liste de l'espace Processes.
5. Cliquez avec le bouton droit de la souris sur **Simulate Post-Place and Route Model**.
6. Choisissez l'onglet **Process Properties** du menu déroulant.

7. Assurez-vous que l'item **Advanced** est sélectionné dans la liste Property display level, dans le bas de la boîte de dialogue.
8. Sélectionnez l'item **ISim properties** dans la liste Category de gauche.
9. Entrez **1ps** dans pour le Simulation Run Time.
10. Cliquez sur **OK**.

Lancez la simulation en double-cliquant sur **Simulate Post-Place and Route Model** avec le bouton de gauche de la souris.

Le logiciel ISim s'ouvre comme illustré à la Figure 60.

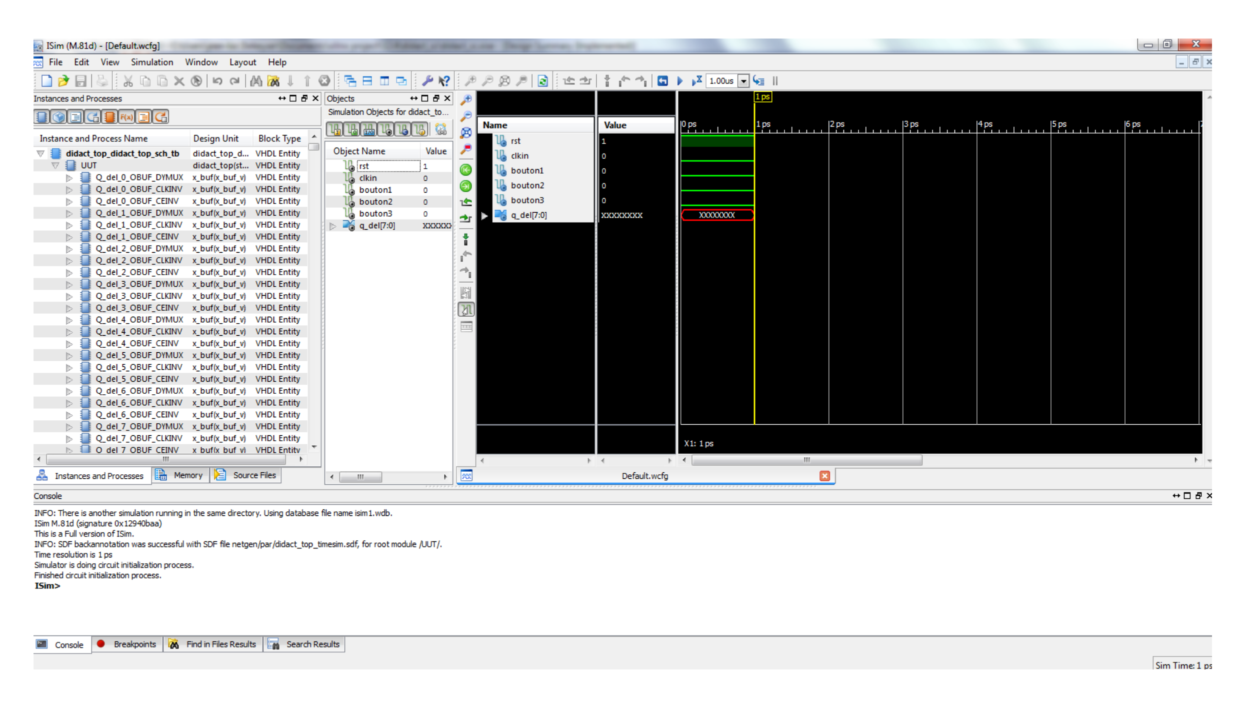


Figure 60. La fenêtre de simulation ISim s'ouvre.

On retrouve l'item UUT dans l'onglet Instances and Processes sous didact_top_tb. Déployer l'item uut et remarquez que les sous modules instanciés sont très différents de ceux montrés à la Figure 48 pour la simulation fonctionnelle. En effet, ISim utilise les modèles Xilinx des composants utilisés pour l'implémentation du projet.

## Configurez la simulation ISim

Configurez ensuite le temps de simulation :

- Entrez **30 ms** dans le Run Length dans la barre d'outils, comme montré à la Figure 50.

## Lancez la simulation

Enfin, lancer la simulation en cliquant sur l'icône **Run** dans la barre d'outils (Figure 50).

La simulation prendra plusieurs minutes pour s'exécuter. Vous pouvez suivre le temps d'exécution annoté dans le bas de la fenêtre ISim.

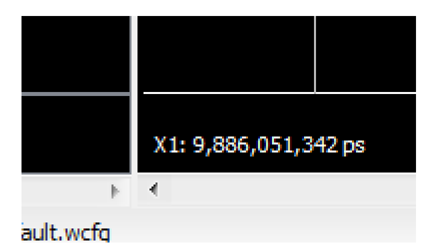



Figure 61. Le temps d'exécution de la simulation est annoté dans la fenêtre.

## Visualisez les résultats

Pour vérifier les résultats de simulation, cliquez d'abord sur l'icône **Zoom full** de la barre d'outils.

La valeur des signaux s'affiche dans la fenêtre Wave pour la totalité du temps de simulation.

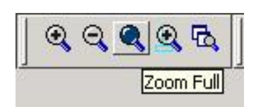



Figure 62. Cliquez sur l'icône Zoom Full pour visualiser la simulation complète sur 30 ms.

Finalement, vérifiez que vos résultats de simulation sont conformes à ceux obtenus avec la simulation fonctionnelle (section Visualisez les résultats).

Fermez le logiciel ISim. Vous avez complété l'étape de simulation avec timings du détecteur de séquence.

Vous êtes prêts à configurer le FPGA Spartan 6 avec le projet :

- Allez au chapitre Configuration de la carte Nexys3 pour configurer le FPGA avec le projet du détecteur de séquence synchrone.

## Configuration de la carte Nexys3 avec ADEPT

Après avoir fait la synthèse, le mapping et le placement & routage du projet du détecteur de séquence synchrone lors de l'étape d'implémentation du projet, vous allez générer le fichier de programmation BIT pour configurer la carte Nexys3.

**Note :** Nous assumons à cette étape que vous avez complétée l'étape préalable d'implémentation du projet détaillée précédemment dans ce didacticiel.

### Création du fichier de programmation

Ouvrer tout d'abord le projet du détecteur de séquence dans le Navigateur ISE, si ce n'est pas déjà fait. Assurez-vous de sélectionner l'onglet **Design** et l'item **Implementation** de la liste **View**, juste au-dessus de l'espace Hierarchy.

Effectuez les configurations préalables suivantes :

1. Sélectionnez **didact_top** dans l'arborescence du projet.
2. Cliquez avec le bouton droit de la souris sur l'item **Generate Programming File**.
3. Choisissez l'onglet **Process Properties** du menu déroulant.
4. Assurez-vous que l'item **Advanced** est sélectionné dans la liste Property display level dans le bas de la boîte de dialogue.
5. Sélectionnez l'item **StartUp Options** dans la liste Category de gauche.
6. Sélectionnez l'item **JTAG Clock** pour le FPGA Start-Up Clock.
7. Cliquez sur **Ok**.

Fabriquez ensuite le fichier de programmation :

8. Dans l'espace Processes, double-cliquez sur l'item **Generate Programming File**.

ISE lance le programme intégré BitGen pour générer le fichier BIT qui servira à programmer le FPGA.

9. Consultez les messages dans la Console et assurez-vous que la création du fichier s'est bien déroulée. Un crochet devrait s'afficher à côté de **Generate Programming File**, comme montré à la Figure 63.

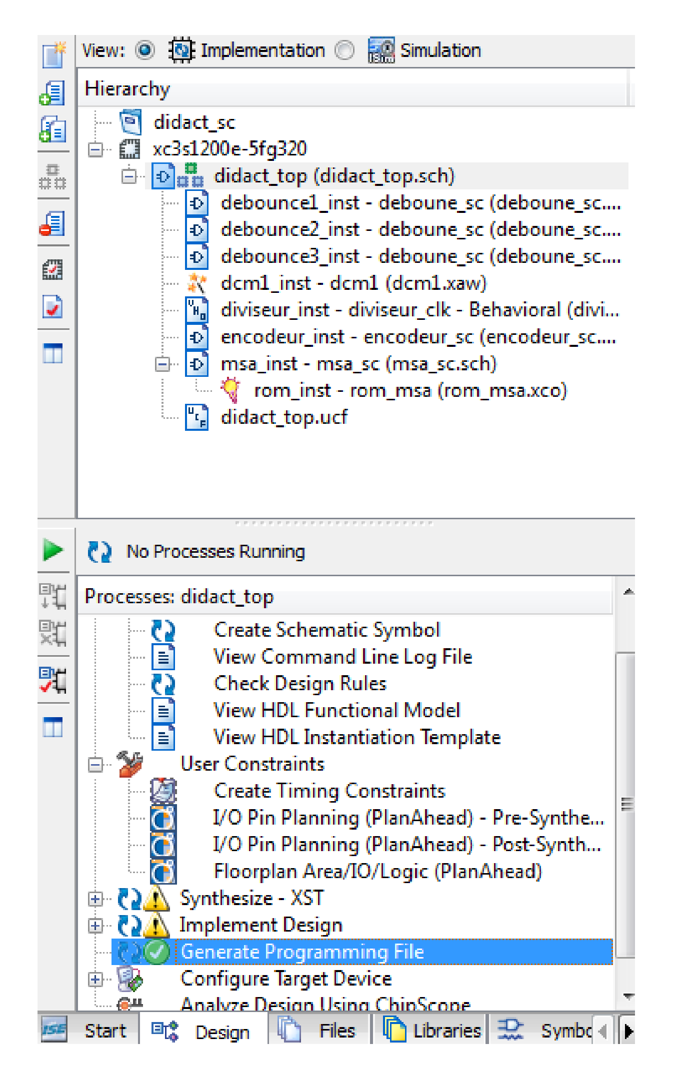



Figure 63. Le fichier de programmation BIT à été généré avec succès.

## Programmez le FPGA en mode JTAG

Configuration  de la carte sans Adept :

Chargement du design sur le FPGA sous Linux, soit en ligne de commande.

- Vérifiez que votre carte FPGA soit branchée et allumée.
- Ouvrez un terminal, seulement 2 commandes sont utiles.
- La commande « djtgcfg enum » : cette commande permet de lister toutes les cartes connectées à l'ordinateur.
- La commande « djtgcfg –d Nexys3 prog –i 0 –f  toplevel.bit »
    - Nexys3: nom de la carte
    - toplevel.bit : bitstream

Vous allez utiliser le logiciel ADEPT pour programmer le FPGA en mode JTAG avec le fichier BIT généré à l'étape précédente. Connectez le câble USB entre votre poste de travail et la prise USB de la carte Nexys3 puis mettez la carte Nexys3 sous tension à l'aide du commutateur SWP (voir la Figure 2 pour situer ce commutateur). Si la carte n'est pas allumée il faut changer le Jumper JP1 afin d'utiliser  USB comme source d'alimentation. Demandez conseil!!!!

**Note :** En mode de programmation JTAG, il est nécessaire de reprogrammer la carte à chaque remise sous tension car le programme du FPGA est recopié dans de la mémoire volatile.

Procédez comme suit pour programmer le FPGA:

Lance le logiciel ADEPT vous obtenez la **Erreur ! Source du renvoi introuvable.**. Il suffit de retrouver le fichier didact_top.bit dans votre espace de travail et ensuite cliquez sur Program.

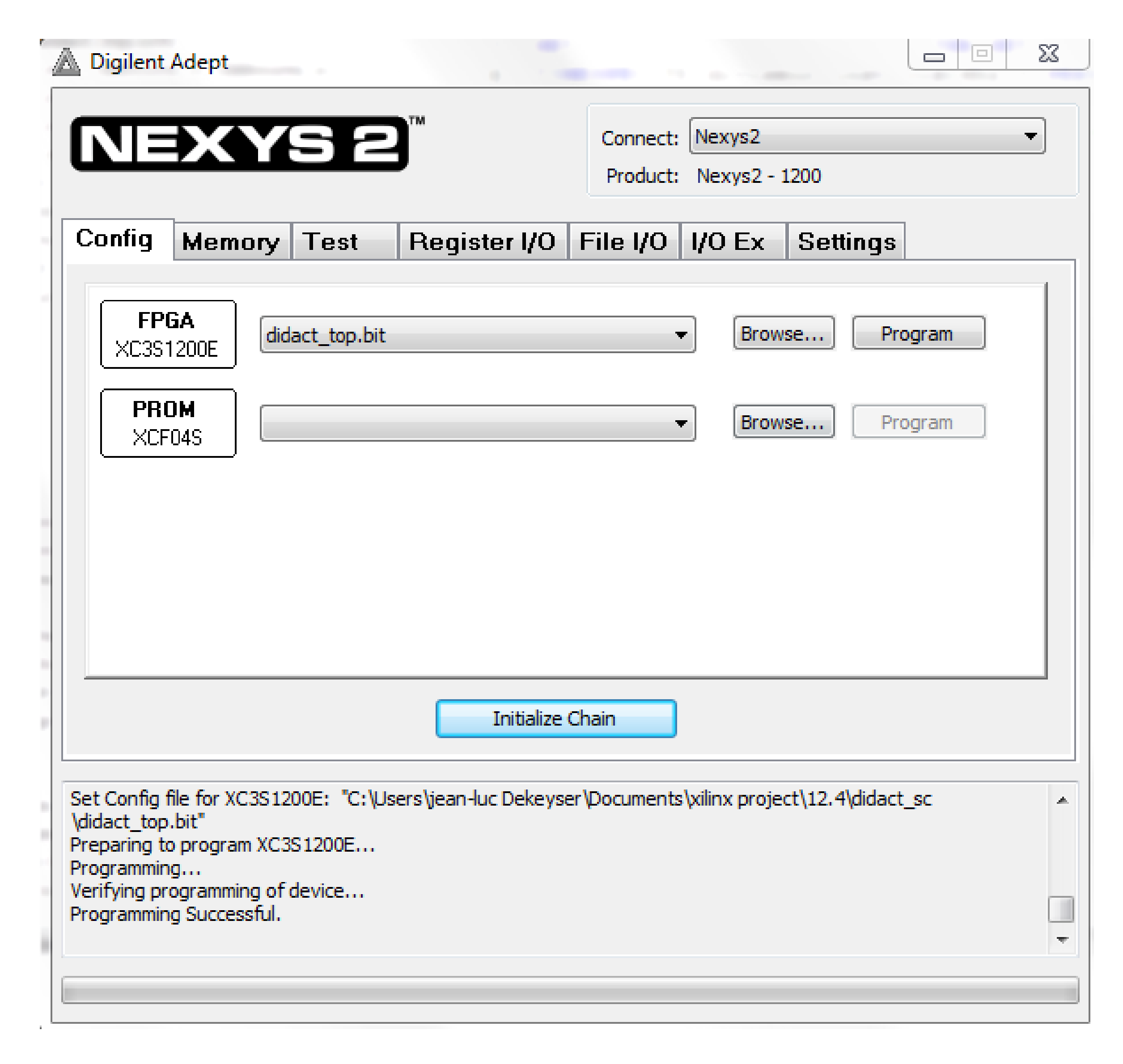



Figure 64. Fenêtre ADEPT pour charger la carte NEXYS2 en mode JTAG.

Le FPGA est maintenant programmé avec le projet du détecteur de séquence. Entrez la séquence en appuyant successivement sur les boutons BTNR, BTNS et BTNL et vérifiez que les Leds LD0 à LD7 s'activent en rafale. En appuyant sur le BTND on peut recommencer la saisie du code.

# Annexe 1

## Description VHDL du module diviseur_clk (fichier diviseur_clk.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity diviseur_clk is
    Port ( clkin : in  STD_LOGIC;
           clk2hz : out  STD_LOGIC;
           clk16hz : out  STD_LOGIC;
           clk2khz : out  STD_LOGIC);
end diviseur_clk;

architecture Behavioral of diviseur_clk is

signal cnt2hz : integer range 0 to 4e6:= 0;
signal cnt16hz : integer range 0 to 5e5:= 0;
signal cnt2khz : integer range 0 to 4e3:= 0;
signal div2hz_temp : std_logic := '0';
signal div16hz_temp : std_logic := '0';
signal div2khz_temp : std_logic := '0';

begin
process (clkin) begin
      if (clkin'event and clkin = '1') then

            -- Diviseur par 4e6
            if cnt2hz >= 4e6 then
                  div2hz_temp <= not(div2hz_temp);
                  cnt2hz <= 0;
            else
                  div2hz_temp <= div2hz_temp;
                  cnt2hz <= cnt2hz + 1;
            end if;
```

```
            clk2hz <= div2hz_temp; -- horloge à 2 Hz

            -- Diviseur par 5e5
            if cnt16hz >= 5e5 then
                  div16hz_temp <= not(div16hz_temp);
                  cnt16hz <= 0;
            else
                  div16hz_temp <= div16hz_temp;
                  cnt16hz <= cnt16hz + 1;
            end if;
            clk16hz <= div16hz_temp; -- horloge à 16 Hz

            -- Diviseur par 4e3
            if cnt2khz >= 4e3 then
                  div2khz_temp <= not(div2khz_temp);
                  cnt2khz <= 0;
            else
                  div2khz_temp <= div2khz_temp;
                  cnt2khz <= cnt2khz + 1;
            end if;
            clk2khz <= div2khz_temp; -- horloge à 2 kHz
      end if;
end process;

end Behavioral;
```

# Annexe 2

## Table de d'entrées/sorties et d'états futurs de la rom_msa

| | (Adresses de la ROM) | | | | | | (Contenu des espaces mémoires) | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Nom | États présents | | | Entrées | | | États futurs | | | Sortie |
| d'état | Q2 | Q1 | Q0 | b1 | b0 | GS | Q2+ | Q1+ | Q0+ | S |
| a | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| b | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| c | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| d | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| e | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| f | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

# Annexe 3

## Fichier d'initialisation du module rom_msa

```
MEMORY_INITIALIZATION_RADIX=2;
MEMORY_INITIALIZATION_VECTOR=
0000
0010
0000
0000
0000
0000
0000
0000
0100
0010
0100
0010
0100
0010
0100
0010
0100
0000
0100
0110
0100
0000
0100
0000
1000
0110
1000
0110
1000
0110
1000
0110
1000
0000
1000
0000
1000
1010
```

```
1000
0000
1011
1011
1011
1011
1011
1011
1011
1011;
```

# Annexe 4

## Description VHDL du module debounce_hdl (fichier debounce_hdl.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity debounce_hdl is
    Port ( sig_in : in  STD_LOGIC;
           clkin : in  STD_LOGIC;
           sig_out : out  STD_LOGIC);
end debounce_hdl;

architecture Behavioral of debounce_hdl is
signal Q1, Q2, Q3 : std_logic;

begin

process(clkin)
begin
   if (clkin'event and clkin = '1') then
       Q1 <= sig_in;
       Q2 <= Q1;
       Q3 <= Q2;
   end if;
end process;

sig_out <= Q1 and Q2 and (not Q3);

end Behavioral;
```

# Annexe 5

## Description VHDL du module msa_hdl (fichier msa_hdl.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity msa_hdl is
    Port ( clkin : in  STD_LOGIC;
           rst : in  STD_LOGIC;
           b0 : in  STD_LOGIC;
           b1 : in  STD_LOGIC;
           gs : in  STD_LOGIC;
           enable_del : out  STD_LOGIC);
end msa_hdl;

architecture Behavioral of msa_hdl is

type etat is (a,b,c,d,e,f);
signal etatpres, etatsuiv : etat;

begin

--registre d'état
xreg: process(rst,clkin)
begin
	if(rst = '1')then
		etatpres <= a;
	elsif(clkin'event and clkin = '1')then
		etatpres <= etatsuiv;
	end if;
end process;

--IFL
xifl: process(etatpres, b1,b0,gs)
begin
	case etatpres is
		when a =>
			if(gs = '1' and b1 = '0' and b0 = '0')then
				etatsuiv <= b;
			else
				etatsuiv <= a;
			end if;
		when b =>
			if(gs = '0')then
				etatsuiv <= c;
			else
				etatsuiv <= b;
			end if;
```

```
			when c =>
				if(gs = '1') then
					if(b1 = '0' and b0 = '1') then
						etatsuiv <= d;
					else
						etatsuiv <= a;
					end if;
				else
					etatsuiv <= c;
				end if;
			when d =>
				if(gs = '0')then
					etatsuiv <= e;
				else
					etatsuiv <= d;
				end if;
			when e =>
				if(gs = '1') then
					if(b1 = '1' and b0 = '0') then
						etatsuiv <= f;
					else
						etatsuiv <= a;
					end if;
				else
					etatsuiv <= e;
				end if;
			when f =>
				etatsuiv <= f;
			when others => etatsuiv <= a;
		end case;
end process;

--OFL
enable_del <= '1' when etatpres = f else '0';

end Behavioral;
```

# Annexe 6

## Fichier VHDL top level didact_top.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity didact_top is
Port ( rst : in  STD_LOGIC;
       clkin : in  STD_LOGIC;
       bouton1 : in  STD_LOGIC;
       bouton2 : in  STD_LOGIC;
       bouton3 : in  STD_LOGIC;
       Q_del : out  STD_LOGIC_VECTOR (7 downto 0));
end didact_top;

architecture Behavioral of didact_top is

component debounce_hdl
port(
       sig_in : in  STD_LOGIC;
       clkin : in  STD_LOGIC;
       sig_out : out  STD_LOGIC);
end component;

COMPONENT dcm1
PORT(
       CLKIN_IN : IN std_logic;
       RST_IN : IN std_logic;
       CLKFX_OUT : OUT std_logic;
       CLKIN_IBUFG_OUT : OUT std_logic;
       CLK0_OUT : OUT std_logic;
       LOCKED_OUT : OUT std_logic
       );
END COMPONENT;

component diviseur_clk
port(
clkin : in  STD_LOGIC;
       clk2hz : buffer  STD_LOGIC;
       clk16hz : buffer  STD_LOGIC;
       clk2khz : buffer  STD_LOGIC);
end component;

component msa_hdl
port(
       clkin : in  STD_LOGIC;
       rst : in  STD_LOGIC;
       b0 : in  STD_LOGIC;
```

```
       b1 : in  STD_LOGIC;
       gs : in  STD_LOGIC;
       enable_del : out  STD_LOGIC);
end component;

signal clk_dcm1 : std_logic;
signal CLKIN_IBUFG_OUT : std_logic;
signal CLK0_OUT : std_logic;
signal LOCKED_OUT : std_logic;
signal b0, b1, gs, enable_del : std_logic;
signal clk2hz,clk16hz,clk2khz : std_logic;
signal debout1,debout2,debout3 : std_logic;
signal shreg : std_logic_vector(7 downto 0);

begin

inst1_debounce: debounce_hdl port map(
       sig_in => bouton1,
       sig_out => debout1,
       clkin => clk2khz
);

inst2_debounce: debounce_hdl port map(
       sig_in => bouton2,
       sig_out => debout2,
       clkin => clk2khz
);

inst3_debounce: debounce_hdl port map(
sig_in => bouton3,
       sig_out => debout3,
       clkin => clk2khz
);

Inst_dcm1: dcm1 PORT MAP(
       CLKIN_IN => clkin,
       RST_IN => rst,
       CLKFX_OUT => clk_dcm1,
       CLKIN_IBUFG_OUT => CLKIN_IBUFG_OUT,
       CLK0_OUT => CLK0_OUT,
       LOCKED_OUT => LOCKED_OUT
);

inst_diviseur_clk: diviseur_clk port map(
       clkin => clk_dcm1,
   clk2hz => clk2hz,
   clk16hz => clk16hz,
   clk2khz => clk2khz
);

Inst_msa_hdl: msa_hdl port map(
       clkin => clk_dcm1,
       rst => rst,
       b0 => b0,
       b1 => b1,
```

```
        gs => gs,
        enable_del => enable_del
);

-- Description de l'encodeur
b0 <= '1' when (debout3 = '0' and debout2 = '1' and debout1 = '0') or
                    (debout3 = '1' and debout2 = '1' and debout1 = '0') else
                '0';
b1 <= '1' when debout3 = '1' and debout2 = '0' and debout1 = '0' else
                '0';

-- Description du Get something
gs <= '1' when debout3 = '1' or debout2 = '1' or debout1 = '1' else
                '0';

-- Description du registre à décalage
xshifreg: process(rst,clk16hz)
begin
        if(rst = '1')then
                shreg <= (others => '0');
        elsif(clk16hz'event and clk16hz = '1')then
                if(enable_del = '1')then
                        shreg(0)<= clk2hz;
                        shreg(7 downto 1) <= shreg(6 downto 0);
                end if;
        end if;
end process;

Q_del <= shreg;

end Behavioral;
```

# Annexe 7

## Fichier testbench didact_top_tb.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;

ENTITY msa_top_tb IS
END msa_top_tb;

ARCHITECTURE behavioral OF didact_top_tb IS

   COMPONENT didact_top
   PORT(      clkin : in  STD_LOGIC;
              rst : in STD_LOGIC;
              bouton1, bouton2,bouton3 : std_logic;
              Q_del : out std_logic_vector(7 downto 0)
               );
   END COMPONENT;

   SIGNAL rst :       STD_LOGIC;
   SIGNAL clkin :     STD_LOGIC:='0';
       signal bouton1, bouton2,bouton3 : std_logic;
       signal Q_del: std_logic_vector(7 downto 0);

BEGIN

   UUT: didact_top PORT MAP(
              rst => rst,
              clkin => clkin,
              bouton1 => bouton1,
              bouton2 => bouton2,
              bouton3 => bouton3,
          Q_del => Q_del);

-- *** Test Bench - User Defined Section ***

-- process pour générer l'horloge
-- une période d'horloge de 20 ns est utilisée
       clkin_gen:process(clkin)
       begin
              clkin <= not clkin after 10 ns;
       end process clkin_gen;

-- process de d'assignation des signaux de test
```

```
   tb : PROCESS
   BEGIN
             rst <='1';
             bouton1 <= '0';
             bouton2 <= '0';
             bouton3 <= '0';
             wait for 100 us;
             rst <= '0';
             bouton1 <= '0';
             bouton2 <= '0';
             bouton3 <= '0';
             wait for 4000 us;
             bouton1 <= '1';
             bouton2 <= '0';
             bouton3 <= '0';
             wait for 4000 us;
             bouton1 <= '0';
             bouton2 <= '0';
             bouton3 <= '0';
             wait for 4000 us;
             bouton1 <= '0';
             bouton2 <= '1';
             bouton3 <= '0';
             wait for 4000 us;
             bouton1 <= '0';
             bouton2 <= '0';
             bouton3 <= '0';
             wait for 4000 us;
             bouton1 <= '0';
             bouton2 <= '0';
             bouton3 <= '1';
             wait for 4000 us;
             bouton1 <= '0';
             bouton2 <= '0';
             bouton3 <= '0';
      WAIT; -- will wait forever
   END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

# Annexe 8

## Fichier de contraintes UCF

```
NET "clkin" TNM_NET = "clkin";
TIMESPEC TS_clkin = PERIOD "clkin" 10 ns HIGH 50 %;

# PlanAhead Generated physical constraints

NET "Q_del[0]" LOC = U16;
NET "Q_del[1]" LOC = V16;
NET "Q_del[2]" LOC = U15;
NET "Q_del[3]" LOC = V15;
NET "Q_del[4]" LOC = M11;
NET "Q_del[5]" LOC = N11;
NET "Q_del[6]" LOC = R11;
NET "Q_del[7]" LOC = T11;
NET "bouton1" LOC = D9;
NET "bouton2" LOC = B8;
NET "bouton3" LOC = C4;
NET "clkin" LOC = V10;
NET "rst" LOC = C9;
```